JBoss Enterprise BRMS Platform 5
Manual del usuario de BRMS

Para desarrolladores de JBoss Rules, escritores de reglas y analistas empresariales.
Edición 5.2.0

Landmann

JBoss Enterprise BRMS Platform 5 Manual del usuario de BRMS

Para desarrolladores de JBoss Rules, escritores de reglas y analistas empresariales.
Edición 5.2.0

Landmann
rlandmann@redhat.com

Legal Notice

Copyright © 2010 Red Hat, Inc..

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Un manual para utilizar JBoss Enterprise BRMS Platform.

Table of Contents

Prefacio **4**
1. Convenciones del Documento 4
1.1. Convenciones tipográficas 4
1.2. Convenciones del documento 5
1.3. Notas y Advertencias 6
2. Cómo obtener ayuda y hacer sus comentarios 6
2.1. ¿Necesita ayuda? 6
2.2. ¡Necesitamos sus comentarios! 7

Capítulo 1. Introducción **8**
1.1. Aspectos nuevos en esta edición 8
1.2. ¿Qué es un BRMS? 8
1.3. ¿Cuándo se debe utilizar un BRMS? 8
1.4. ¿Quién utiliza BRMS? 9
1.5. Resumen de las funcionalidades 9

Capítulo 2. Arquitectura **10**
2.1. Componentes re-utilizables 10
2.2. Versionamiento y almacenamiento 11

Capítulo 3. Manual de inicio rápido **12**
3.1. Navegadores soportados 12
3.2. ¿BRMS o Guvnor? 13
3.3. Configuración inicial 13
3.4. Escritura de reglas 13
3.5. Búsqueda 13
3.6. Implementación 14

Capítulo 4. Conceptos de BRMS **15**
4.1. Las reglas son activos 15
4.2. Categorización 15
4.3. Autoría de reglas 17
4.3.1. El editor de activos 17
4.3.2. Reglas empresariales con el editor dirigido 17
4.3.3. Anatomía de una regla 18
4.3.4. Relevancia 20
4.3.5. Listas desplegables dirigidas por el usuario 20
4.3.6. Incremento con declaraciones DSL 21
4.3.7. Reglas DSL 21
4.3.8. Hoja de cálculo de las tablas de decisión 22
4.3.9. tablas de decisión dirigidas (basadas en la web) 22
4.3.9.1. Componentes principales 23
4.3.9.2. Configuración de columnas 23
4.3.9.2.1. Columnas de funcionalidades 24
4.3.9.2.2. Columnas de atributos 24
4.3.9.2.3. Columnas de meta-datos 24
4.3.9.2.4. Columnas de condiciones 24
4.3.9.2.5. Columnas de acción 24
4.3.9.3. Definición de reglas 25
4.3.9.4. Fusión de celdas 25
4.3.9.5. Agrupamiento de celdas 25
4.3.9.6. Operaciones Otherwise 26
4.3.10. Plantillas de reglas 26

4.3.10.1. Definición de datos de la plantilla 29
4.3.10.1.1. Fusión de celdas 30
4.3.10.1.2. Agrupamiento de celdas 31
4.3.10.2. DRL generado 32
4.3.11. Flujo de reglas 33
4.3.12. Reglas técnicas (DRL) 33
4.3.13. Funciones 34
4.3.14. Enumeraciones de datos (Configuraciones de la lista desplegable) 34
4.3.15. Conceptos avanzados de enumeración 35
4.4. Administración del estatus 36
4.5. Administración de paquetes 37
4.5.1. Importación de paquetes drl 40
4.6. Administración de versiones 41
4.7. Administración de implementación 41
4.8. Navegación del repositorio y ubicación de reglas 42

Capítulo 5. El modelo de hechos (el modelo de objetos) 44
5.1. El área global 44
5.2. Modelo JAR 44
5.3. Modelo declarativo 46
5.3.1. Consumo de un modelo declarativo de Java 47

Capítulo 6. grupos de trabajo 49
6.1. Verificación de restricciones de campos 50

Capítulo 7. La perspectiva del usuario empresarial 52

Capítulo 8. Integración de reglas con sus aplicaciones 53
8.1. El agente de conocimiento 53
8.2. Implementación manual 54
8.3. WebDAV 55
8.3.1. WebDav y caracteres especiales 55
8.4. URLs 55

Capítulo 9. Buzón de entrada y comentarios 57
9.1. Comentarios 57
9.2. Buzón de entrada 57

Capítulo 10. Integración de JBoss Developer Studio 59
10.1. Sinopsis de las funcionalidades 59
10.2. Asistente de conexión Guvnor 60
10.3. Explorador del repositorio Guvnor 62
10.4. Copias locales de archivos Guvnor 64
10.5. Acciones para recursos locales Guvnor 66
10.6. Importación de recursos del repositorio Guvnor 69
10.7. Preferencias plug-in de Guvnor 71

Capítulo 11. Configuración de clientes para la firma de paquetes de reglas 73

Historial de revisiones 75

Prefacio

1. Convenciones del Documento

Este manual utiliza varias convenciones para resaltar algunas palabras y frases y llamar la atención sobre ciertas partes específicas de información.

En ediciones PDF y de papel, este manual utiliza tipos de letra procedentes de Liberation Fonts. Liberation Fonts también se utilizan en ediciones de HTML si están instalados en su sistema. Si no, se muestran tipografías alternativas pero equivalentes. Nota: Red Hat Enterprise Linux 5 y siguientes incluyen Liberation Fonts predeterminadas.

1.1. Convenciones tipográficas

Se utilizan cuatro convenciones tipográficas para llamar la atención sobre palabras o frases específicas. Dichas convenciones y las circunstancias en que se aplican son las siguientes:

`Negrita monoespaciado`

Utilizado para resaltar la entrada del sistema, incluyendo los comandos de shell, nombres de archivos y rutas. También sirve para resaltar teclas y combinaciones de teclas. Por ejemplo:

> Para ver el contenido del archivo **`my_next_bestselling_novel`** en su directorio actual de trabajo, escriba el comando **`cat my_next_bestselling_novel`** en el intérprete de comandos de shell y pulse **`Enter`** para ejecutar el comando.

El ejemplo anterior incluye un nombre de archivo, un comando de shell y una tecla . Todo se presenta en negrita-monoespaciado y distinguible gracias al contexto.

Las combinaciones de teclas se pueden distinguir de las individuales con el signo más que conecta cada partee de la combinación de tecla. Por ejemplo:

> Pulse **`Enter`** para ejecutar el comando.
>
> Pulse **`Ctrl`**+**`Alt`**+**`F2`** para pasar a una terminal virtual.

El primer ejemplo resalta una tecla particular a pulsar. El segundo ejemplo, resalta una combinación de teclas: un set de tres teclas pulsadas simultáneamente.

Si se discute el código fuente, los nombres de las clase, los métodos, las funciones, los nombres de variables y valores de retorno mencionados dentro de un párrafo serán presentados en **`Negrita-monoespaciado`**. Por ejemplo:

> Las clases de archivo relacionadas incluyen **`filename`** para sistema de archivos, **`file`** para archivos y **`dir`** para directorios. Cada clase tiene su propio conjunto asociado de permisos.

Negrita proporcional

Esta denota palabras o frases encontradas en un sistema, incluyendo nombres de aplicación, texto de cuadro de diálogo, botones etiquetados, etiquetas de cajilla de verificación y botón de radio; títulos de menú y títulos del submenú. Por ejemplo:

> Seleccione **Sistema** → **Preferencias** → **Ratón** desde la barra del menú principal para lanzar **Preferencias de ratón**. En la pestaña de **`Botones`**, seleccione la cajilla de **`ratón`**

> **de mano izquierda** y luego haga clic en **Cerrar** para cambiar el botón principal del ratón de la izquierda a la derecha (adecuando el ratón para la mano izquierda).
>
> Para insertar un carácter especial en un archivo **gedit**, seleccione **Aplicaciones** → **Accesorios** → **Mapa de caracteres** de la barra del menú. Luego, seleccione **Búsqueda** → **Buscar...** de la barra del menú de **Mapa de caracteres**, escriba el nombre del carácter en el campo de **Búsqueda** y haga clic en **Siguiente**. El carácter que buscó será resaltado en la **Tabla de caracteres**. Haga doble clic en ese carácter resaltado para colocarlo en el campo de **Texto a copiar** y luego haga clic en el botón **Copiar**. Ahora regrese al documento y elija **Modificar** → **Pegar** de la barra de menú de **gedit**.

El texto anterior incluye nombres de aplicación; nombres y elementos del menú de todo el sistema; nombres de menú de aplicaciones específicas y botones y texto hallados dentro de una interfaz gráfica de usuario, todos presentados en negrita proporcional y distinguibles por contexto.

Itálicas-negrita monoespaciado o ***Itálicas-negrita proporcional***

Ya sea negrita monoespaciado o negrita proporcional, la adición de itálicas indica texto reemplazable o variable. Las itálicas denotan texto que usted no escribe literalmente o texto mostrado que cambia dependiendo de la circunstancia. Por ejemplo:

> Para conectar a una máquina remota utilizando ssh, teclee **ssh *nombre de usuario@dominio.nombre*** en un intérprete de comandos de shell. Si la máquina remota es **example.com** y su nombre de usuario en esa máquina es john, teclee **ssh john@example.com**.
>
> El comando **mount -o remount *file-system*** remonta el sistema de archivo llamado. Por ejemplo, para volver a montar el sistema de archivo **/home**, el comando es **mount -o remount /home**.
>
> Para ver la versión de un paquete actualmente instalado, utilice el comando **rpm -q *paquete***. Éste entregará el resultado siguiente: ***paquete-versión-lanzamiento***.

Observe que las palabras resaltadas en itálicas — nombre de usuario, dominio.nombre, sistema de archivo, paquete, versión y lanzamiento. Cada palabra es un marcador de posición, ya sea de texto a ingresar cuando se ejecuta un comando o para un texto ejecutado por el sistema.

Aparte del uso estándar para presentar el título de un trabajo, las itálicas denotan el primer uso de un término nuevo e importante. Por ejemplo:

> Publican es un sistema de publicación de *DocBook*.

1.2. Convenciones del documento

Los mensajes de salida de la terminal o fragmentos de código fuente se distinguen visualmente del texto circundante.

Los mensajes de salida enviados a una terminal se muestran en **romano monoespaciado** y se presentan así:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Los listados de código fuente también se muestran en **romano monoespaciado**, pero se presentan y resaltan de la siguiente manera:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notas y Advertencias

Finalmente, utilizamos tres estilos visuales para llamar la atención sobre la información que de otro modo se podría pasar por alto.

Nota

Una nota es una sugerencia, atajo o enfoque alternativo para una tarea determinada. Ignorar una nota no debería tener consecuencias negativas, pero podría perderse de algunos trucos que pueden facilitarle las cosas.

Importante

Los cuadros con el título de importante dan detalles de cosas que se pueden pasar por alto fácilmente: cambios de configuración únicamente aplicables a la sesión actual, o servicios que necesitan reiniciarse antes de que se aplique una actualización. Ignorar estos cuadros no ocasionará pérdida de datos, pero puede causar enfado y frustración.

Aviso

Las advertencias no deben ignorarse. Ignorarlas muy probablemente ocasionará pérdida de datos.

2. Cómo obtener ayuda y hacer sus comentarios

2.1. ¿Necesita ayuda?

Si encuentra dificultades con alguno de los procedimientos descritos en este documento, visite el Portal del cliente de Red Hat en http://access.redhat.com. A través del portal del cliente, usted podrá:

- buscar o navegar a través de la base de artículos de soporte técnico sobre productos de Red Hat.
- enviar un caso de soporte a Servicios de Soporte Global de Red Hat (GSS)
- acceder a otra documentación del producto.

Red Hat alberga una lista grande de correos electrónicos para discutir sobre software de Red Hat y tecnología. Encontrará un listado de las listas de correo disponibles al público en https://www.redhat.com/mailman/listinfo. Haga clic en el nombre de la lista a la que quiera suscribirse o para acceder a los archivos de listados.

2.2. ¡Necesitamos sus comentarios!

Si encuentra algun error o si se le ocurre una manera de mejorar este manual, nos encantaría escuchar sus sugerencias. Complete un reporte en Bugzilla frente al producto **JBoss Enterprise BRMS Platform 5** y el componente **doc-BRMS_User_Guide**. El siguiente enlace le llevará a un reporte de error ya completado para este producto: http://bugzilla.redhat.com/.

Llene la siguiente plantilla en el campo de **Description** de Bugzilla. Sea tan especifico como le sea posible al describir el problema, esto ayudará a asegurarnos de que lo podemos solucionar rápidamente.

```
URL del documento:


Número de la sección y nombre:


Describa el problema:


Sugerencias para mejorar:


Información adicional:
```

Asegúrese de darnos su nombre para poder darle todo el crédito por reportar el problema.

Capítulo 1. Introducción

1.1. Aspectos nuevos en esta edición

Tabla 1.1. Aspectos nuevos en esta edición

Funcionalidad	Cambio
Sección 4.3.9, “ tablas de decisión dirigidas (basadas en la web)”	Se agregaron detalles adicionales sobre las tablas de decisiones
Sección 4.3.10, “Plantillas de reglas”	Se agregó una sección sobre las plantillas de reglas.
Capítulo 5, *El modelo de hechos (el modelo de objetos)*	Se agregó información adicional sobre el modelo de hechos.
Capítulo 6, *grupos de trabajo*	Se agregó un nuevo capítulo sobre los grupos de trabajo.
Sección 8.3.1, “WebDav y caracteres especiales”	Se agregaron instrucciones para utilizar caracteres multibyte en toda la WebDAV.

1.2. ¿Qué es un BRMS?

JBoss Enterprise BRMS es un sistema de administración de reglas empresariales.

JBoss Enterprise BRMS proporciona herramientas para administrar reglas empresariales de software en un entorno con múltiples usuarios. Es el punto único de verdad para sus reglas empresariales, lo que permite que los cambios tengan lugar de manera controlada, utilizando una interfaz amigable para el usuario.

JBoss Enterprise BRMS Platform es una solución del lado del servidor basada en JBoss Rules para la administración, almacenamiento, modificación e implementación de reglas y otros activos de JBoss Rules. También se proporciona una interfaz de usuario basado en la web así como integración para JBoss Developer Studio y otros entornos de desarrollo integrados basados en Eclipse.

Importante

En varios lugares de JBoss Enterprise BRMS Platform y de su documentación se refiere a Guvnor. JBosss Guvnor es el nombre del proyecto de código abierto sobre el cual se ha construído JBoss Enterprise BRMS Platform. Las referencias a Guvnor permanecen en la API, las URLs y en las herramientas de JBoss Developer Studio (consulte Capítulo 10, *Integración de JBoss Developer Studio*).

1.3. ¿Cuándo se debe utilizar un BRMS?

Un sistema de administración de reglas empresariales es útil en los siguientes escenarios:

1. Se necesita administrar un sistema para implementación y modificación de reglas.
2. Múltiples usuarios con diferentes niveles de habilidad necesitan acceder y modificar las reglas.
3. No existe una infraestructura para administrar las reglas.
4. Hay muchas reglas "empresariales" que se deben administrar de manera opuesta a las reglas

técnicas que serán parte de una aplicación.

1.4. ¿Quién utiliza BRMS?

Las siguientes personas con los siguientes cargos en su empresa son principalmente los que utilizarán el sistema de administración de las reglas empresariales:

- Analistas empresariales
- Expertos en reglas
- Desarrolladores
- Administradores de reglas

JBoss Enterprise BRMS Platform permite asignar diferentes roles a diferentes usuarios para controlar los activos y las funcionalidades que se exponen.

1.5. Resumen de las funcionalidades

- El soporte para múltiples idiomas en la interfaz de usuario web BRMS actualmente es inglés de Estados Unidos, japonés y chino simplificado.
- Varios tipos de editores de reglas (gráfico y basado en texto)
- Control de versiones (para activos históricos)
- Categorización
- Construcción e implementación
- Almacenar múltiples "activos" de reglas como un solo paquete.

Capítulo 2. Arquitectura

Este capítulo cubre los aspectos técnicos del diseño de la plataforma de JBoss Enterprise BRMS. No es una lectura obligatoria para los usuarios de la aplicación BRMS. Esta dirigido a los desarrolladores que integran la plataforma JBoss Enterprise BRMS con otros sistemas.

Figura 2.1, “Diagrama arquitectónico” muestra los componentes más importantes del sistema, la manera en que se integran y se implementan.

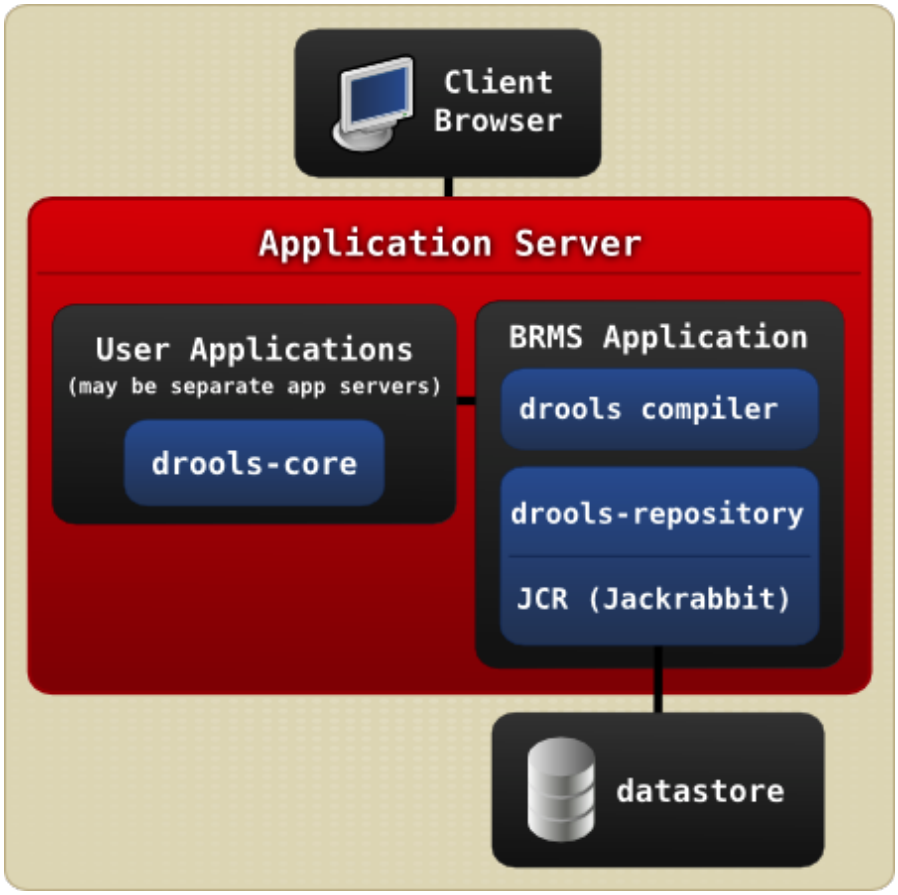


Figura 2.1. Diagrama arquitectónico

BRMS se implementa como un WAR, el cual proporciona interfaces de usuario a través de la web y paquetes binarios por medio de URLs. Utiliza el estándar JSR-170 para almacenamiento de datos (JCR). JBoss Seam se utiliza como el marco de trabajo del componente y GWT se utiliza como el kit de herramientas para construir la interfaz de usuario web basada en ajax.

2.1. Componentes re-utilizables

BRMS usa una interfaz de servicio para separar la interfaz gráfica del usuario de la funcionalidad de segundo plano. En este caso el segundo plano incluye el repositorio de activos así como las especificaciones del compilador para tratar con las reglas.

La interfaz principal es **`RepositoryService`**, la cual se implementa en **`ServiceImplementation`**. El plano principal ajax GWT le habla a esta interfaz usando el mecanismo de callback asincrónica de GWT. El archivo de configuración Seam es **`components.xml`**.

Esta interfaz de servicio la pueden re-utilizar componentes opcionales o de primer plano.

La interfaz de usuario GWT se puede volver a utilizar ya que GWT es solo una página html: **`Guvnor.html`**. Para aquellos familiarizados con GWT, cada una de las *funcionalidades* se puede utilizar por separado. La clase **`JBRMSFeature`** y las clases que la implementan (en teoría pueden ser autónomas) contienen información adicional.

2.2. Versionamiento y almacenamiento

En la base de datos se almacenan versiones de activos junto con los datos.

Cuando se crean *tomas de pantalla* se realizan copias de todo el paquete en un lugar separado en la base de datos JCR.

Para aquellos familiarizados con JCR y jackrabbit, los archivos **`.cnd`** se encuentran en la fuente para las definiciones de tipo de nodos ya que algunos desean verlas. Un paquete es un *folder* y cada activo es un archivo: un archivo puede ser textual o puede tener un anexo binario.

Capítulo 3. Manual de inicio rápido

Esta sección proporcionará un tour rápido de funcionalidades de la plataforma JBoss Enterprise BRMS. Se asume que la plataforma BRMS y su repositorio se encuentran instalados y configurados correctamente.

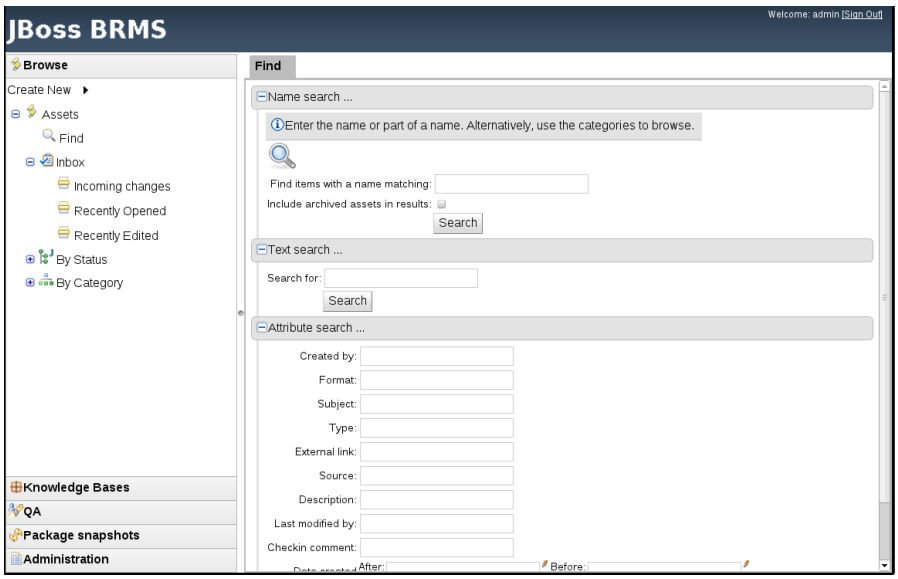


Figura 3.1. Interfaz de usuario web de JBoss Enterprise BRMS Platform

Figura 3.1, “Interfaz de usuario web de JBoss Enterprise BRMS Platform ” muestra las áreas principales de JBoss Enterprise BRMS Platform.

El panel de navegación en la izquiera proporciona un rápido acceso a todas las áreas principales de la interfaz de usuario web BRMS. Estas áreas son:

- Info: Esta es la pantalla inicial con enlaces a los recursos.
- Reglas: Esta es la categoría y la perspectiva del usuario empresarial.
- Paquete: Aquí es donde se configuran y administran los paquetes de conocimiento.
- Implementación: aquí es donde se administran las tomas de pantalla de la implementación.
- Admin: funciones administrativas (categorías, estatus, importación y exportación).

3.1. Navegadores soportados

Los navegadores soportados para ver la interfaz de usuario web BRMS se pueden ver en Tabla 3.1, “Navegadores soportados”

Tabla 3.1. Navegadores soportados

Sistema operativo	Navegadores
RHEL 5.x y posteriores	FireFox 3.0+
Microsoft Windows	FireFox 3.0+
Microsoft Windows	Internet Explorer 7+
Mac OSX 10.x	FireFox 3.0+
Mac OSX 10.x	Safari 4 y 5

3.2. ¿BRMS o Guvnor?

En versiones previas de Drools "BRMS" con frecuencia se utilizaba para referirse a la interfaz web para las funcionalidades de administración drools. Hoy en día usamos BRMS para referirse a "todo el paquete" - el tiempo de ejecución, las herramientas web, etc - pero en algunos casos "BRMS" se puede entender como la consola web Guvnor y las herramientas asociadas.

3.3. Configuración inicial

Se requiere una configuración inicial para la primera vez. La primera vez que el servidor inicia se creará un repositorio vacío y luego tome los siguientes pasos:

- Si es un repositorio nuevo vaya a "Admin" y seleccione "Manage Categories"

  Agregue unas pocas categorías (note que las categorías son sólo con el fin de la clasificación).
- Las reglas necesitan un modelo de hechos (también conocido como un modelo de objetos) con el cual trabajar. Desde la funcionalidad "Package Management" tal como se esperaría se puede crear un nuevo paquete de conocimiento. Los paquetes deben tener nombres significativos sin espacios.
- Para cargar un modelo use un archivo **`.jar`** que contenga el modelo de hechos (API) que estará utilñizando en sus reglas y en su código. Cuando se encuentre en la pantalla "Model Editor" puede cargar un archivo **`.jar`**. Para hacer esto seleccione el nombre del paquete de la lista que creó en el paso anterior.
- Ahora modifique la configuración del paquete que acaba de crear con el fin de importar los tipos de hechos que ha cargado (estas son las declaraciones de importación). Guarde los cambios.
- En este momento el paquete está configurado y está listo para utilizarse.

  Note que también puede importar un paquete DRL (Drools Rule Language) ya existente y las reglas se almacenarán en el repositorio como activos individuales.

3.4. Escritura de reglas

- Una vez que tenga configurados una categoría y un paquete puede empezar a escribir reglas.
- Hay múltiples formatos de reglas pero BRMS los considera "activos".
- Puede crear una regla haciendo clic en el logo de la cabecera y posteriormente introducir el nombre.
- También tendrá que escoger una categoría. Las categorías proporcionan una manera de ver las reglas separado de los paquetes de conocimiento (de hecho puede hacer que las reglas aparezcan en múltiples paquetes de conocimiento). Puede que le sea útil el considerarlo como una etiqueta.
- Seleccione los formatos "Business Rule (Guided Editor)".
- Esto abrirá un modelador de reglas, el cual es un editor dirigido. Puede agregar y modificar condiciones y acciones con base en el modelo que se esté utilizando en el paquete actual. También estará disponible cualquier plantilla de oraciones DSL configurada para el paquete.
- Cuando haya terminado de modificar las reglas puede guardar los cambios o también puede escoger el validar o "ver fuente" (para la fuente efectiva).
- También puede agregar o borrar categorias del editor de reglas y puede modificar otros atributos tal como la documentación, (si no está seguro de qué hacer entonces escriba un documento con lenguaje normal que describa la regla y guárdelo. Después lo puede utilizar como una plantilla).

3.5. Búsqueda

Con el fin de navegar en el sistema puede utilizar la funcionalidad de las reglas, (la cual muestra las

cosas agrupadas por categorías) o puede utilizar la funcionalidad para paquetes y ver por paquetes (o tipo de reglas). Si conoce el nombre o parte del nombre de un activo también puede usar el "Quick Find." Para utilizarlo empiece a escribir el nombre de la regla y BRMS retornará una lista de coincidencias mientras escribe.

3.6. Implementación

- Después de modificar algunas reglas en un paquete puede hacer clic en la funcionalidad "Package", abra el paquete que desee y constrúyalo.
- Si el proceso de construcción tiene éxito podrá descargar un archivo de paquete binario, el cual luego se puede implementar en un sistema en tiempo de ejecución.
- También puede realizar una "toma de pantalla" de un paquete para la implementación. Esto congela el paquete en ese justo momento de manera que ninguno de los cambios actuales no lo afectan. Esto también hace disponible el paquete en una URL de la siguiente forma: **`http://<your server>/jboss-brms/org.drools.guvnor.Guvnor/packages/<packageName>/<snapshotName>`**

Capítulo 4. Conceptos de BRMS

4.1. Las reglas son activos

Un "activo" es cualquier cosa que se pueda almacenar como una versión en el repositorio. Esto incluye reglas, tablas de decisión, modelos, pruebas y DSLs.

Nota

El término "regla" con frecuencia se utiliza muy vagamente. A veces se utiliza incorrectamente para referirse a activos que de hecho no son reglas.

4.2. Categorización

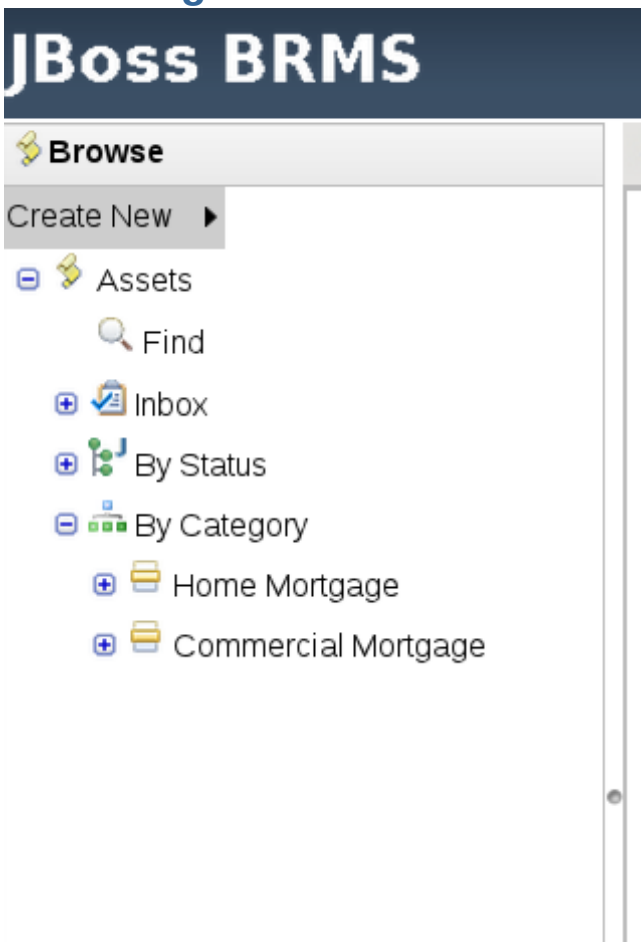


Figura 4.1. Categorias

Las categorias permiten etiquetar las reglas (activos) con cualquier número de agrupamientos que usted defina. Esto significa que luego puede ver una lista de reglas que coinciden con una categoría específica. Las reglas pueden pertenecer a cualquier número de categorias. En el diagrama anterior puede ver que se presenta como una carpeta/explorador como una vista de activos. Los nombres pueden ser como los quiera y los definen el administrador BRMS (también puede borrar /agregar nuevas categorias; solo puede borrarlas si actualmente no se encuentran en uso).

Generalmente las categorias se crean con nombres significativos que coinciden con el área empresarial a la cual aplica la regla (por lo tanto si la regla aplica a múltiples áreas se pueden adjuntar múltiples categorias). Las categorias también se pueden utilizar para "etiquetar" reglas como parte de su ciclo de vida, por ejemplo para marcar algo como "borrador" o "por revisar".

Figura 4.2. Los activos pueden tener múltiples categorías

La vista anterior muestra la categoría Editor/Vista que se puede ver cuando abre un activo. En este ejemplo puede ver que el activo pertenece a dos categorías. Esto significa que cuando cualquiera de las categorías se utiliza para mostrar una lista de activos entonces verá ese activo (el botón The view above shows the Category Editor/Viewer that is seen when you open an asset. In this example, you can see the asset belongs to two categories. This means that when either category is used to show a list of assets, you will see that asset. (The "+" y el ícnono de la basura se utiliza para agregar y borrar cosas adicionales respectivamente).

En el ejemplo anterior, la primera categoría ("Home") se encuentra en el nivel "superior". La segunda, "Mortage/Eligibility" todavía es una sola categoría, pero está anidada ya que las categorias son jerárquicas. En otras palabras, hay una categoría llamada "Mortage", la cual contiene una categoría llamada "Eligibility". La pantalla lo muestra como "Mortage/Eligibility." Como lo puede ver es análogo a la estructura del directorio que tiene en el sistema de archivos de su disco duro aunque las reglas pueden aparecer en múltiples lugares.

Cuando abre un activo con el fin de verlo o editarlo verá una lista de categorías a las cuales pertenece.Si realiza un cambio (borrar o agregar una categoría) será necesario que guarde el activo y esta acción creará un nuevo objeto en le historial de versiones. El cambio de categorías de una regla no afecta su ejecución.

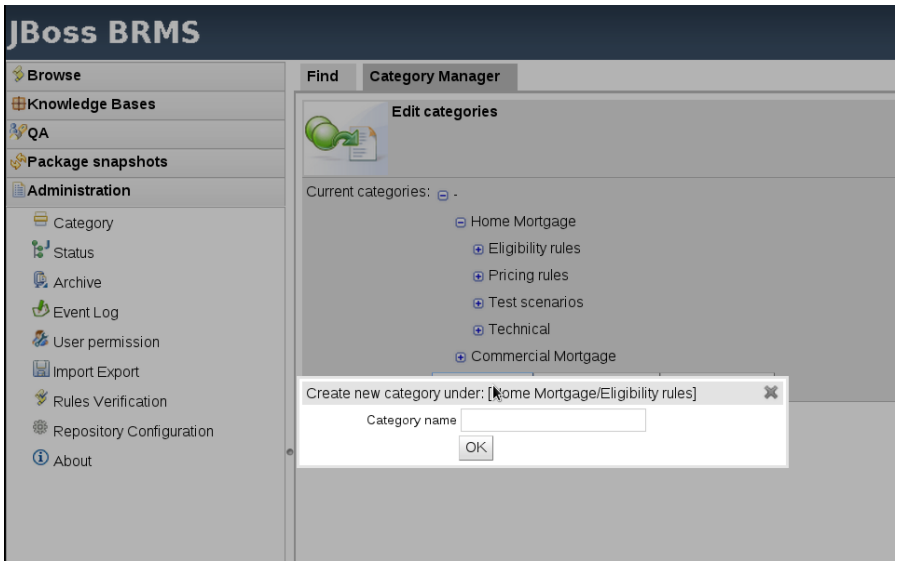


Figura 4.3. Creación de categorías

La vista anterior muestra la pantalla de administración para la creación de categorías. No hay categorías por defecto en el sistema. Ya que las categorías pueden ser jerárquicas debe seleccionar la categoría "padre" para la cual quiere crear una sub-categoría. Desde aquí también se pueden borrar

las categorías (pero solo si no están en uso por parte de cualquiera de los activos actuales).

Nota

Como regla general, un activo solo debe pertenecer a una o dos categorías a la vez. Las categorías son críticas en casos en donde tiene números grandes de reglas. Las jerarquías no necesitan ser muy profundas. Red Hat las diseñó para ayudarle a descomponer reglas/activos en partes administrables.

4.3. Autoría de reglas

La plataforma BRMS soporta varios formatos de activos (regla). Aquí se describen las que son clave. Algunas de estas se cubren en otras partes del manual así que no vamos a repetir esos detalles.

4.3.1. El editor de activos

Las opciones disponibles en el editor de activos dependerán del tipo de activo que se está editando. El editor de activos se utilizar para modificar reglas y se puede utilizar para agregar, borrar y re-ordenar condiciones, restricciones, acciones, campos y opciones. El editor de activos también le permite al usuario el trabajar con grupos, validay y verificar reglas así como ver la fuente de reglas.

Vea los tips sobre las herramientas en el editor de activos para obtener mayores detalles.

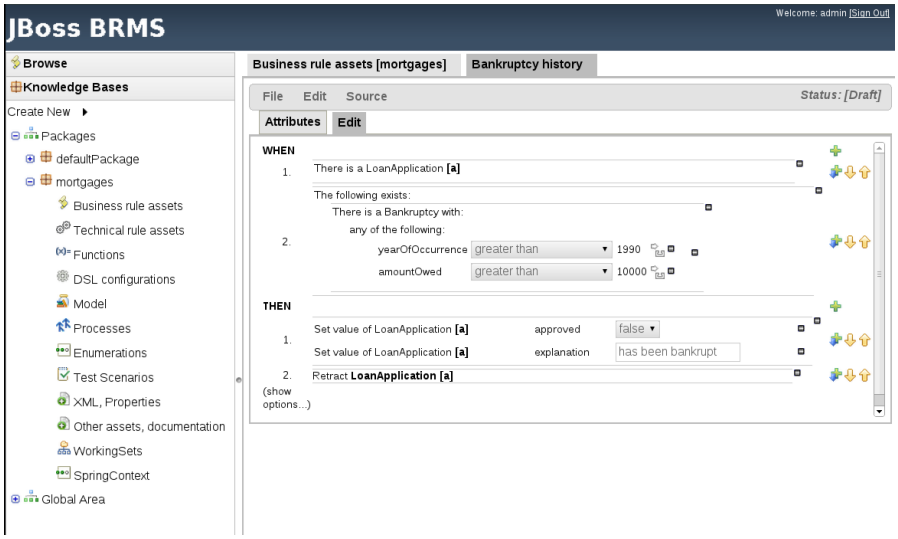


Figura 4.4. Vista del editor de activos

4.3.2. Reglas empresariales con el editor dirigido

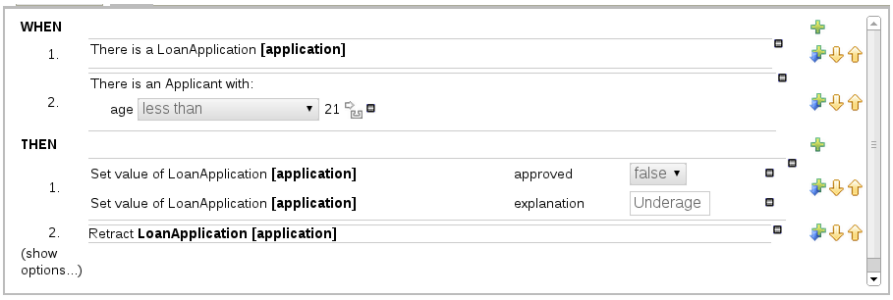
Al modificar reglas el editor de activos también se conoce como el editor dirigido. El **`editor dirigido`** se utiliza para modificar reglas en el formato del lenguaje de reglas empresariales (BRL del inglés Business Rules Language). El editor dirigido le pide a los usuarios entradas con base en el modelo de objetos de la regla que se está modificando.

El acceso al paquete se debe configurar antes de poder utilizar el editor dirigido BRL.

Nota

Yambién hay un editor dirigido que se encuentra en el plug-in de Eclipse. La mayoría de los detalles en esta sección también aplican.

Ejemplo 4.1. El editor dirigido


WHEN
1. There is a LoanApplication [application]
2. There is an Applicant with:
age less than 21
THEN
1. Set value of LoanApplication [application] approved false
Set value of LoanApplication [application] explanation Underage
2. Retract LoanApplication [application]
(show options...)

4.3.3. Anatomía de una regla

Una regla tiene múltiples partes:

- When

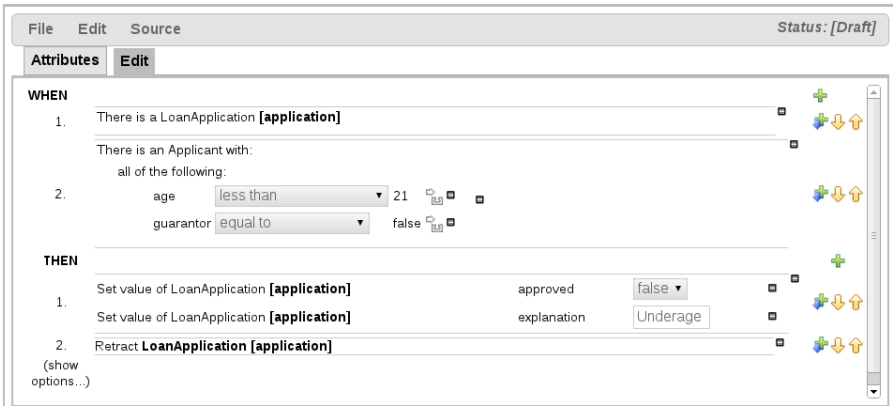
  La parte *When* de la regla es la condición con la que se debe cumplir. Por ejemplo en Ejemplo 4.1, “El editor dirigido”, la sección del when de la reglas es: When the applicant is under 21 years old.

- Then

  La parte *Then* de la regla es la acción que se va a realizar cuando se ha cumplido con la parte condicional de la regla. Por ejemplo, en Ejemplo 4.1, “El editor dirigido”, la parte del then de la regla es: Then decline the loan because the applicant is under age.

Con el editor dirigido es posible agregar más condiciones al When (o la parte condicional) de la regla y se pueden añadir más acciones al Then (o acción) de la regla. Por ejemplo, si un candidato menor de 21 años tuviera un fiador para una aplicación de préstamo es posible que el banco decida aprobar la aplicación de préstamo. Para modelar esto en Ejemplo 4.1, “El editor dirigido” sería necesario el modificar el When para incluir la condición del fiador.

Ejemplo 4.2. Modificación de reglas


File Edit Source Status: [Draft]
Attributes Edit
WHEN
1. There is a LoanApplication [application]
There is an Applicant with:
all of the following:
2. age less than 21
guarantor equal to false
THEN
1. Set value of LoanApplication [application] approved false
Set value of LoanApplication [application] explanation Underage
2. Retract LoanApplication [application]
(show options...)

Para agregar un fiador a la condición primero es necesario el agregar el campo **`guarantor`** (fiador) al tipo de hecho de la aplicación para el modelo de crédito hipotecario.

Procedimiento 4.1. Agregar un campo a un tipo de hecho

1. **Seleccionar el modelo**

   Seleccione **`Knowledge Bases`** del panel de navegación en el lado izquierdo de la pantalla. Expanda el paquete que contiene el modelo y seleccione **`model`**.

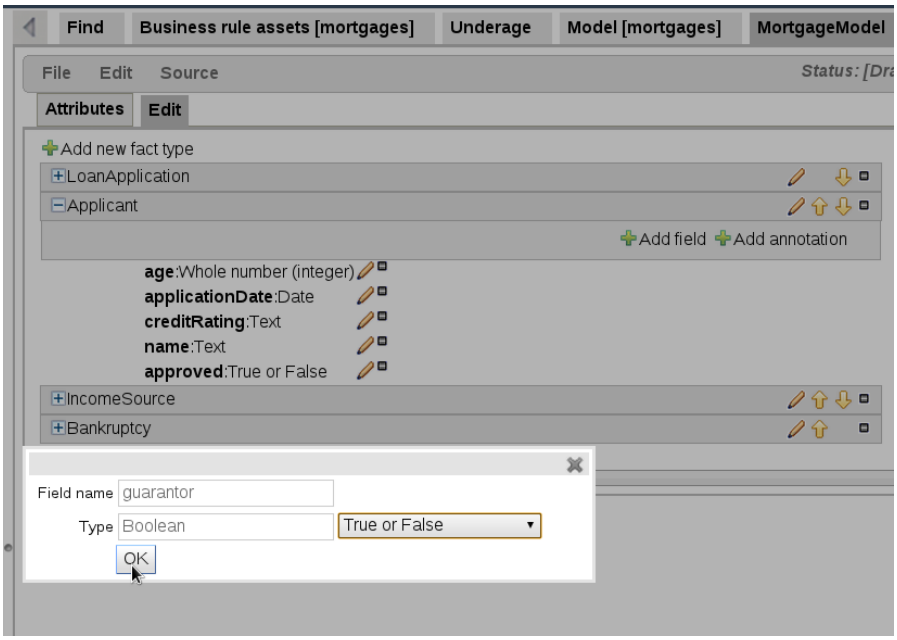
   Abra el modelo de la lista haciendo clic en **`open`**.

2. **Agregue el campo**

   Despliegue el tipo de hecho haciendo clic en el signo más justo al lado y seleccione **`Add Field`**.

3. **Introduzca los detalles del campo**

   Agregue los detalles en la ventana que aparece. En este caso introduzca *fiador* en el campo **`Field name`** y seleccione **`True or False`** del menú desplegable **`Type`**.



   Guarde los cambios realizados al modelo seleccionando **`File`** y **`Save changes`**.

Ya que se agregó el campo del fiador al tipo de hecho del aplicante es posible modificar la regla para incluir un fiador.

Procedimiento 4.2. Agregar restricciones a las reglas

1. **Borrar la restricción actual**

   Al modificar una regla existente puede ser necesafrio el borrar la restricción actual. Esto asegura que se seleccione el operador lógico correcto (And u Or) al tratar con múltiples restricciones.

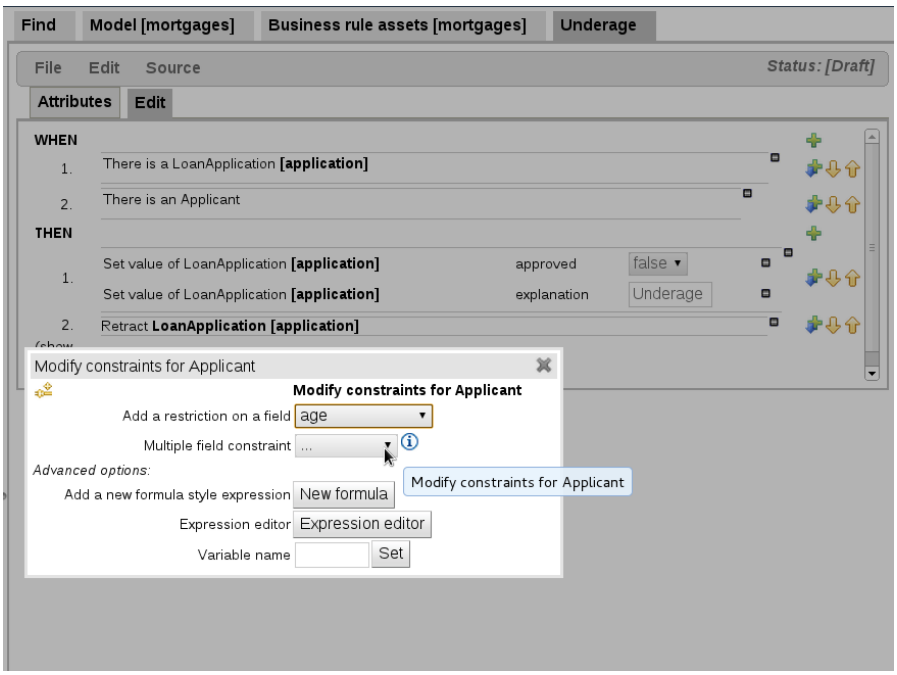
   En este caso haga clic en el símbolo de menos al lado de la condición de la edad.

2. **Modificación de restricciones**

   Abra **`Modify Constraints Dialogue`** haciendo clic en el texto *There is an Applicant*.

   Seleccione la restricción del menú desplegable **`Add a restriction on a field`**. En este caso seleccione **`Age`**.

   Del menú deplegable **`Multiple field constraints`** seleccione **`All of (and)`**.



3. **Especifique la primera restricción**

   Haga clic en el texto *all of the following* para abrir la ventana **`Add fields to this constraint`** y agregue la restricción. En este caso seleccione **`Age`**.

   Del menú desplegable seleccione **`less than`** y modifique el valor literal haciendo clic en el ícono del lápiz, seleccione **`literal`** y luego haga clic en **`Value`** para introducir el valor apropiado, en este caso 21.

4. **Especifique la segunda restricción**

   Haga clic en el texto *all of the following* para abrir la ventana **`Add fields to this constraint`** y agregue la restricción. Esta vez seleccione **`guarantor`**.

   Del nuevo menú desplegable seleccione **`equal to`** y modifique el valor literal haciendo clic sobre el ícono del lápiz, seleccione **`literal`** y luego haga clic en **`true`** para modificar el valor y seleccione falso.

4.3.4. Relevancia

La relevancia de una regla también se puede modificar desde el editor dirigido bajo opciones. La relevancia es un valor numérico que representa la prioridad de la regla.

4.3.5. Listas desplegables dirigidas por el usuario

Note que es posible el limitar los valores ce campos con solo cosas en una lista pre-configurada. Esta lista se configura como parte del paquete (se utiliza una enumeración de datos para poblarlo con valores). Estos valores pueden ser una lista fija o por ejemplo, se pueden cargar desde una base de datos. Esto es útil para los códigos y otros campos en donde hay valores establecidos. También es posible el tener lo que se puede ver en la pantalla en una lista desplegable y que sea diferente al valor (o código) utilizado en una regla. Consulte la sección sobre "Enumeración de datos" para obtener

mayor información sobre la manera en que se configuran.

4.3.6. Incremento con declaraciones DSL

Si el paquete al cual pertenece la regla tiene una configuración DSL, cuando agrega condiciones o acciones, proporcionará una lista de "declaraciones DSL" de donde puede escoger (cuando escoja una agregará una fila a la regla), por medio de la cual los valores especificados por DSL provienen de un usuario y luego se presentará una casilla de modificación (se parece a una forma). Observe que esto es opcional y hay otro editor DSL.

Importante

El editor dirigido actualmente no soporta el grupo completo de funcionalidades DSL. Solo se soportan las secciones **`[when]`** y **`[then]`** de la configuración DSL.

El siguiente diagrama muestra las declaraciones DSL en acción en el editor dirigido:

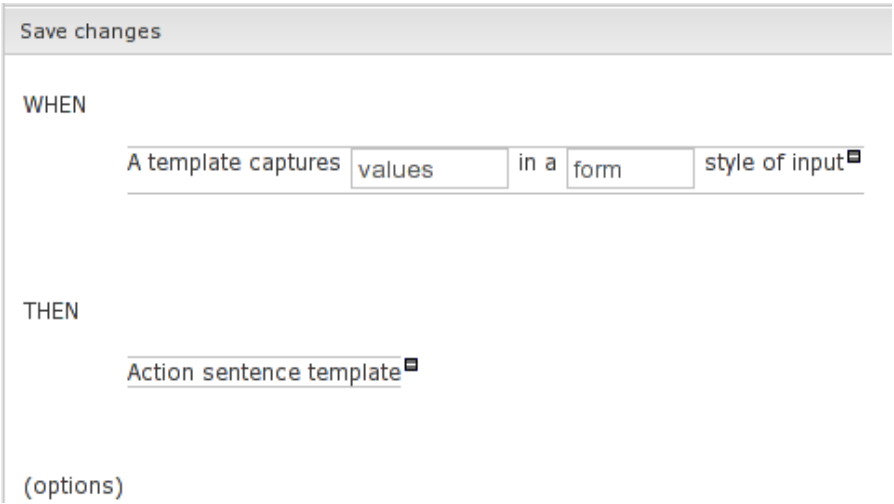


Figura 4.5. DSL en el editor dirigido

4.3.7. Reglas DSL

Las reglas DSL son reglas textuales que utilizan un activo de configuración del lenguaje para controlar su apariencia.

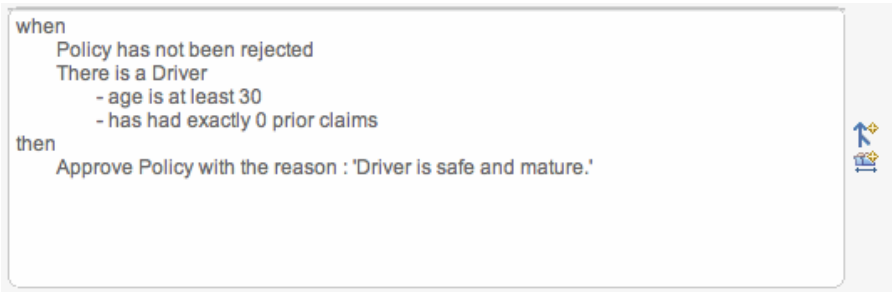

when
Policy has not been rejected
There is a Driver
- age is at least 30
- has had exactly 0 prior claims
then
Approve Policy with the reason : 'Driver is safe and mature.'

Figura 4.6. Regla DSL

Una regla DSL es una sola regla. En la imagen anterior puede ver un editor de texto. Puede utilizar los

íconos a la derecha para proporcionar listas de condiciones y acciones de las cuales escoger (de otra manera presione las teclas control y espacio al mismo tiempo para que la lista aparezca).

4.3.8. Hoja de cálculo de las tablas de decisión

Se pueden almacenar múltiples reglas en una hoja de cálculo de Microsoft Excel (con formato de archivo **`.xls`**), por medio del cual cada fila representa una regla. Los detalles de la hoja de cálculo no los abordamos en este capítulo.

Figura 4.7. Tabla de decisiones en la hoja de cálculo

Para utilizar una hoja de cálculo cargue un archivo **`.xls`** como se puede ver en Figura 4.7, “Tabla de decisiones en la hoja de cálculo”. También puede descargar la versión actual de la hoja de cálculo de la misma ventana. Con el fin de crear una nueva tabla de decisiones debe iniciar el "Rule Wizard" - el asistente de reglas-, el cual contiene una opción para este procedimiento y luego puede cargar el archivo **`.xls`**.

4.3.9. tablas de decisión dirigidas (basadas en la web)

La funcionalidad de las tablas de decisiones dirigidas permite modificar las tablas de decisiones en la web. Esto funciona de manera similar al editor dirigido realizando una introspección de los hechos y campos disponibles para dirigir la creación de una tabla de decisiones. Se pueden definir atributos de reglas, meta-datos, condiciones y acciones en una formato tabular, lo cual facilita la entrada rápida de grupos grandes de reglas relacionadas. Las reglas de las tablas de decisiones basadas en la web se compilan en DRL como todos los otros activos de reglas.

Las celdas se pueden seleccionar de varias maneras:

- Se puede hacer doble clic en celdas individuales y aparece un editor correspondiente la tipo de datos subyacente. Se pueden seleccionar grupos de celdas en la misma columna haciendo clic en la primera y arrastrando el cursos o haciendo clic en la primera y haciendo clic sobre la extención del rango requerido presionando la tecla shift.
- Opcionalmente se pueden utilizar las teclas para manejar el cursor y navegar en la tabla. El presionar la tecla enter hará que aparezca el editor correspondiente. El rango se puede seleccionar presionando la tecla shift y extendiendo el rango con las teclas que mueven el cursor.

Se puede modificar el tamaño de las columnas colocándose sobre el divisor correspondiente en el encabezado de la tabla. El curso del ratón cambiará y luego se puede modificar el ancho de la columna para hacerla más grande o más pequeña.

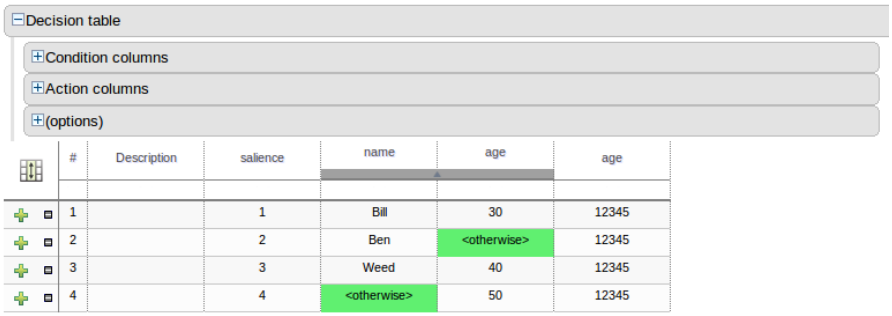


Figura 4.8. Tabla de decisiones

4.3.9.1. Componentes principales

La tabla de decisiones se divide en dos secciones principales:

- La sección superior permite definir columnas de la tabla representando atributos de reglas, metadatos, condiciones y acciones.
- La sección inferior contiene la tabla en sí en donde filas individuales definen reglas separadas.

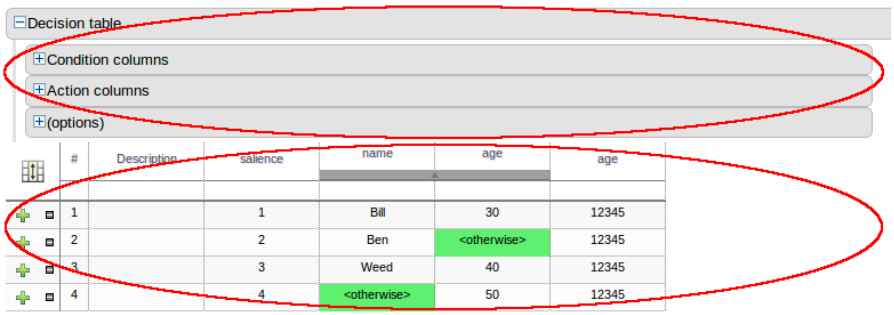


Figura 4.9. Componentes principales

4.3.9.2. Configuración de columnas

Las columnas pueden tener los siguientes tipos de restricción:

- Literal

  El valor en la celda se comparará con el campo utilizando el operador.
- Formula

  La expresioó en la celda se evaluará y luego se comparará con el campo.
- Predicado

  No se necesita ningún campo, la expresión se evaluará como verdadera o falsa.

Puede establecer un valor predeterminado pero normalmente si no hay un valor en la celda entonces esa restricción no aplica.

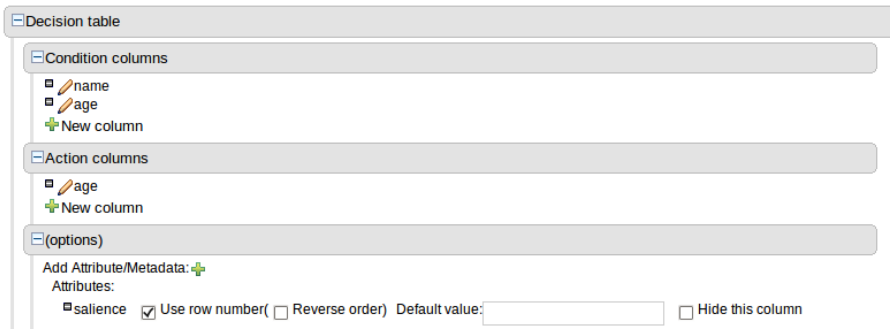


Figura 4.10. Configuración de columnas

4.3.9.2.1. Columnas de funcionalidades

Por defecto se proporcionan dos columnas que contienen el número de la regla y la descripción.

4.3.9.2.2. Columnas de atributos

Se pueden agregar cero o más columnas de atributos que representan cualquiera de los atributos de reglas DRL. Se brinda un pseudo-atributo adicional en el editor de la tabla dirigida de decisiones para "negar" una regla. Este atributo permite negar reglas completas. Por ejemplo como se puede ver la siguiente regla simple se puede negar.

```
when
  $c : Cheese( name == "Cheddar" )
then
  ...
end
```

```
when
  not Cheese( name == "Cheddar" )
then
  ...
end
```

4.3.9.2.3. Columnas de meta-datos

Se pueden definir cero o más columnas de meta-datos, cada una representa la anotación de meta-datos normal en las reglas DRL.

4.3.9.2.4. Columnas de condiciones

Las condiciones representan patrones de hechos definidos en el lado izquierdo o en la parte "when" de una regla. Para definir una columna de condición debe definir un enlace a una clase modelo o debe seleccionar una que se haya definido previamente. Puede escoger el negar el patrón. Una vez que se haya completado esto puede definir restricciones de campos. Si dos o más columnas se definen utilizando el mismo enlace de patrón de hechos, las restricciones de campos se convierten en restricciones de campo compuestas en el mismo patrón. Si define múltiples enlaces para una sola clase modelo cada enlace se vuelve un modelo separado en el lado derecho de la regla.

4.3.9.2.5. Columnas de acción

Las columnas de acción se pueden definir para realizar operaciones simples en hechos enlazados dentro de la memoria de trabajo de la máquina de reglas o para crear hechos completamente nuevos. Los nuevos hechos se pueden incluir lógicamente en la memoria de trabajo de la máquina de reglas y

por lo tanto está sujeta a un mantenimiento de verdad como siempre. Consulte *JBoss Rules Reference Guide* para obtener mayor información sombre el mantenimiento de verdades e inserciones lógicas.

4.3.9.3. Definición de reglas

Esta sección permite definir reglas individuales utilizando las columnas definidas anteriormente.

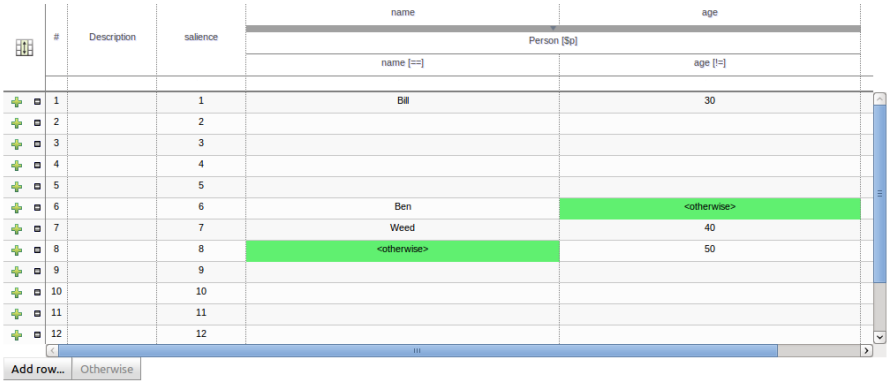


Figura 4.11. Definición de reglas

4.3.9.4. Fusión de celdas

El ícono en la parte superior izquierda de la tabla de decisiones prende y apaga la fusión de celdas. Cuando las celdas se fusionan, las que se encuentran en la misma columna con valores idénticos se fusionan en una sola celda. Esto simplifica el cambio de valor de múltiples celdas que compartían el nismo valor original. Cuando las celdas se fusionan también obtienen un ícono en la parte superior izquierda de la celda que permite agrupar las filas que abarcan la celda fusionada.

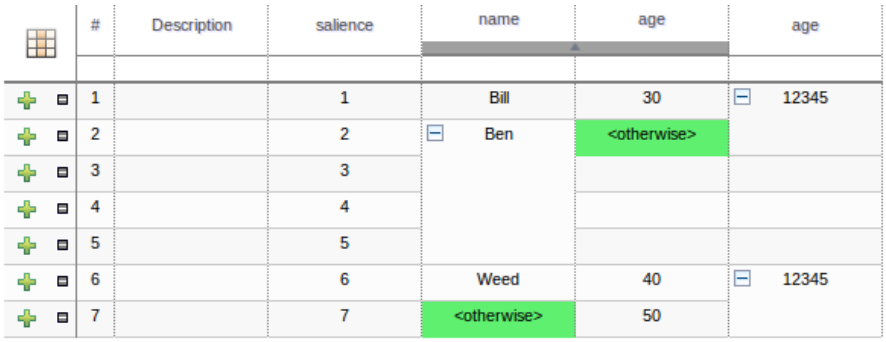
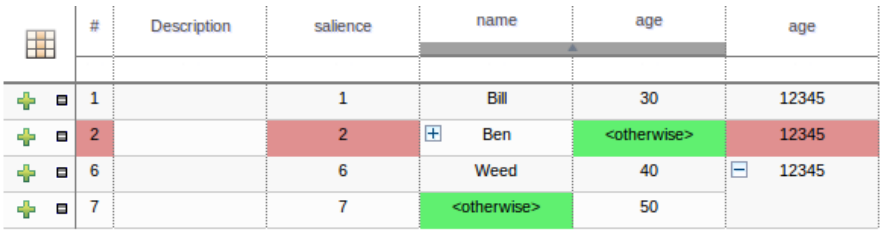


Figura 4.12. Fusión de celdas

4.3.9.5. Agrupamiento de celdas

Las celdas que se han fusionado se pueden plegar en una sola fila. El hacer clic en el ícono [+\-] en la parte superior izquierda de una celda fusionada pliega las filas correspondientes en una sola entrada. Las celdas en otras columnas que abarcan las filas plegadas que tienen valores idénticos se muestran sin ningún cambio. Las celdas en otras columnas que abarcan las filas plegadas que tienen valores diferentes se resaltan y presentan el primer valor.



#	Description	salience	name	age	age
1		1	Bill	30	12345
2		2	Ben	<otherwise>	12345
6		6	Weed	40	12345
7		7	<otherwise>	50	

Figura 4.13. Agrupamiento de celdas

Cuando se altera el valor de una celda agrupada entonces también se actualizan los valores de las celdas que se han plegado.

4.3.9.6. Operaciones Otherwise

Las columnas de condiciones definidas con valores literales que usan los operadores de igualdad **==** o desigualdad **!=** pueden tomar ventaja del valor especial de la celda de la tabla de decisiones **otherwise**. Este valor especial permite definir una regla que coincida con todos los valores no definidos explícitamente en todas las otras reglas definidas en la tabla. Esto se ilustra mejor con un ejemplo:

```
when
  Cheese( name not in ("Cheddar", "Edam", "Brie") )
  ...
then
  ...
end
```

```
when
  Cheese( name in ("Cheddar", "Edam", "Brie") )
  ...
then
  ...
end
```

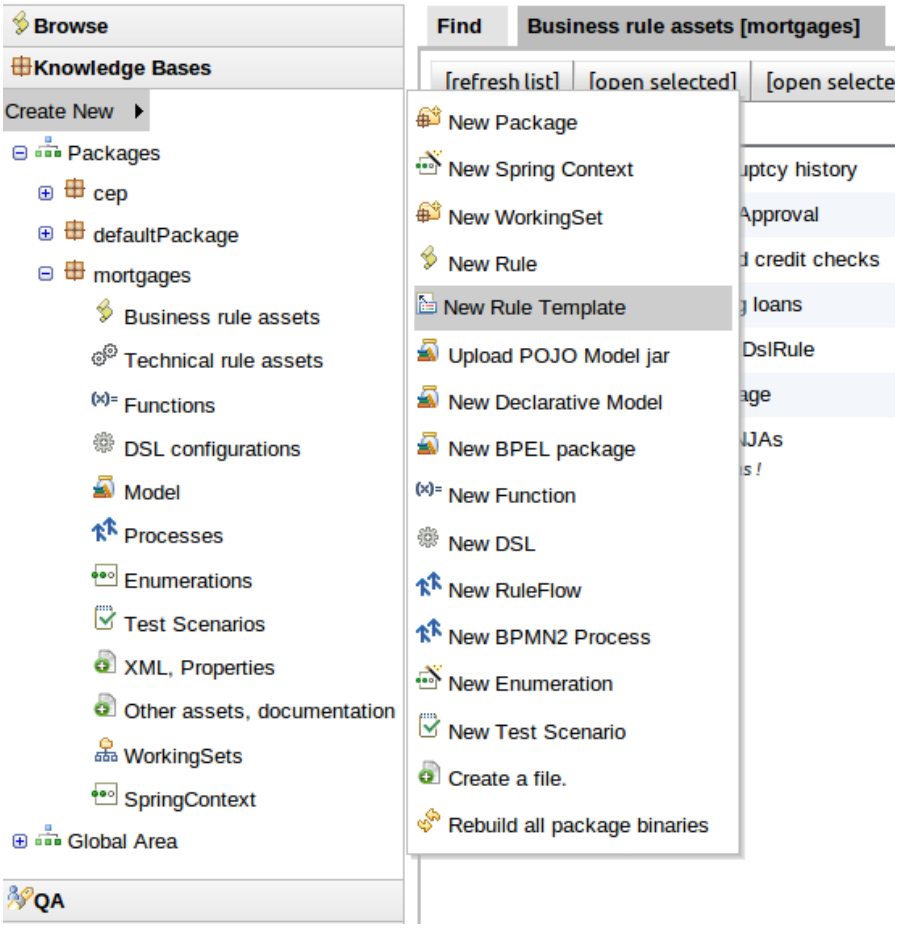
4.3.10. Plantillas de reglas

Las plantillas de reglas le permiten al usuario el definir una estructura de reglas con reservas de espacio para loa valores que se interpolan desde una tabla de datos. También se puede seguir utilizando valores literales, formulas y expresiones.

Las plantillas de reglas con frecuencia se pueden utilizar como una opción para las tablas de decisiones.

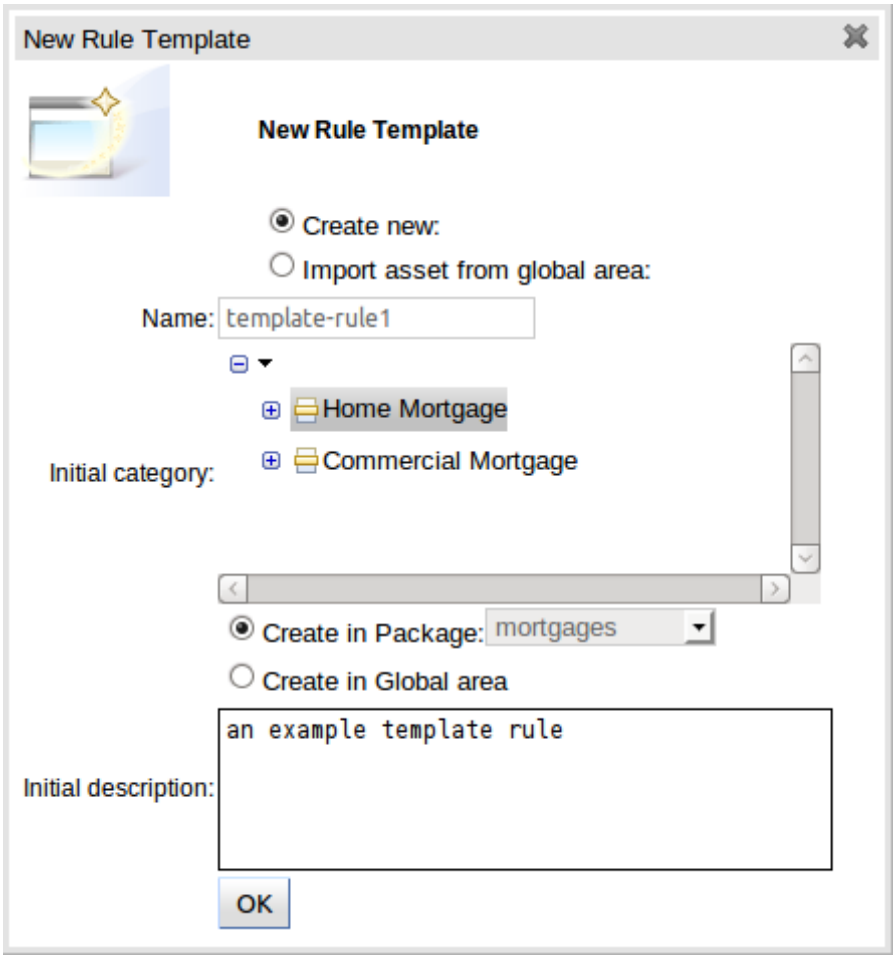
Procedimiento 4.3. Creación de plantillas de reglas

1. **Creación del activo de plantilllas de reglas**
   Del menú **Knowledge Bases** seleccione **Create New** y luego seleccione **New Rule Template**.



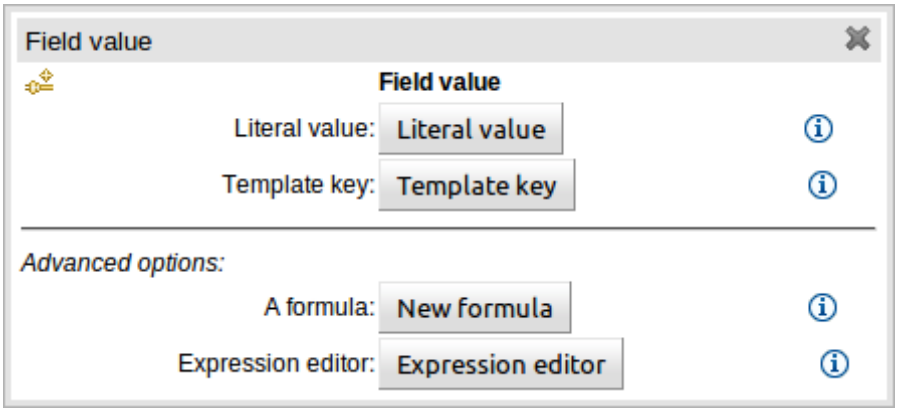
2. **Definición del activo**

   Introduzca un nombre, una categoría y una descripción para la plantilla.



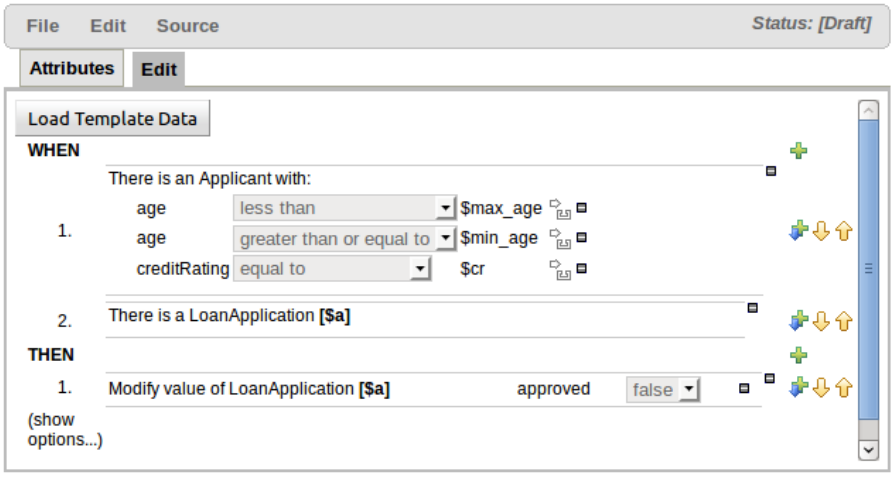
3. **Definición de la plantilla**

   Use el editor dirigido para construir la regla. **`Template keys`** son reservas de espacio dentro de la restricción del campo y las secciones de acción. Se pueden seguir utilizando los valores literales, las fórmulas y expresiones tal como en el editor dirigido estándar.



Ejemplo 4.3, “Plantilla de ejemplo” ilustra una regla simple que se ha definido con una llave de plantilla para la edad máxima del aplicante, su edad mínima y su calificación de crédito. Las llaves de la plantilla se han definido como "$max_age", "$min_age" y "$cr" respectivamente.

Ejemplo 4.3. Plantilla de ejemplo



4.3.10.1. Definición de datos de la plantilla

Cuando complete la definición de su plantilla de reglas necesita introducir los datos que se utilizarán para interpolar las reservas de espacio de la "llave de la plantilla". BRMS brinda la funcionalidad para introducir datos en una cuadrícula flexible dentro de la pantalla del editor dirigido. El editor dirigido se puede iniciar presionando el botón **`Load Template Data`** en la pantalla del editor dirigido.

La cuadrícula de datos de la plantilla de reglas es muy flexible con diferentes editores emergentes para los tipos de datos de los campos subyacentes, Se puede modificar el tamaño de las columnas y también se pueden organizar y las celdas se pueden fusiinar y agrupar para facilitar la entrada rápida de datos.

Una fila de datos interpola las reservas de espacio **`Template Key`** para una sola regla de tal manera que una fila se convierte en una regla.

Nota

Si se dejan celdas en blanco entonces no se genera la regla para la fila aplicable.

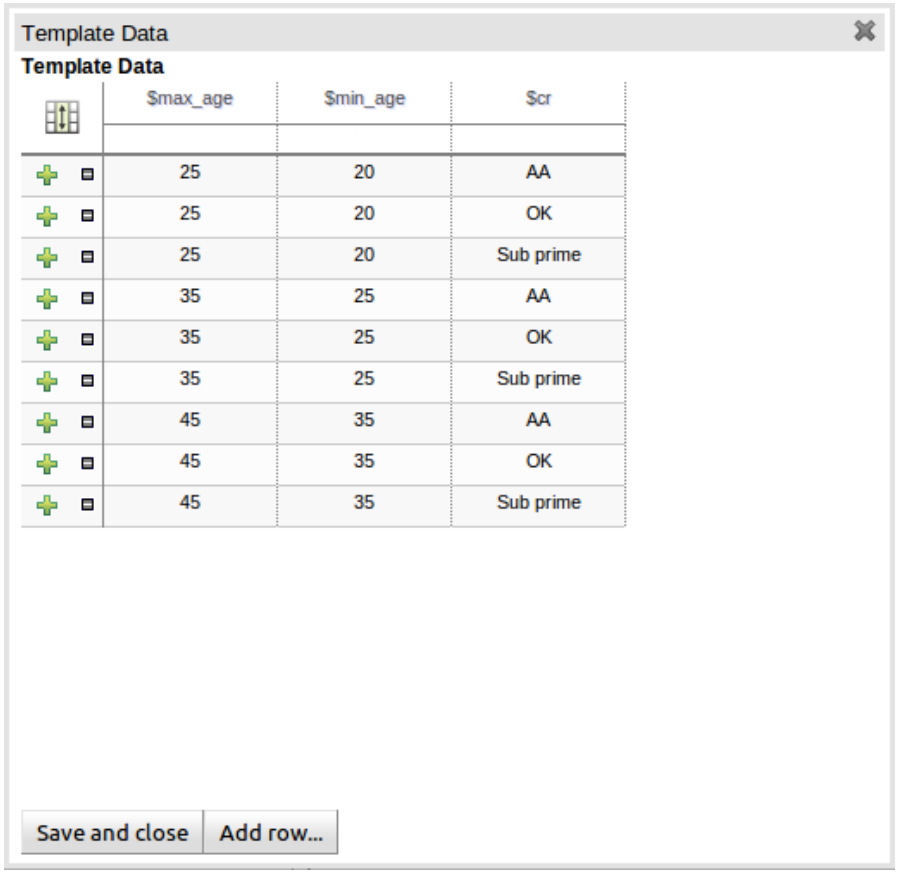


Figura 4.14. Plantilla de la cuadrícula de datos

4.3.10.1.1. Fusión de celdas

El ícono en la parte superior izquierda de la cuadrícula prende y apaga la fusión de celdas. Cuando las celdas se fusionan, las que se encuentran en la misma columna con valores idénticos se fusionan en una sola celda. Esto simplifica el cambio de valor de múltiples celdas que compartían el mismo valor original. Cuando las celdas se fusionan también obtienen un ícono en la parte superior izquierda de la celda que permite agrupar las filas que abarcan la celda fusionada.

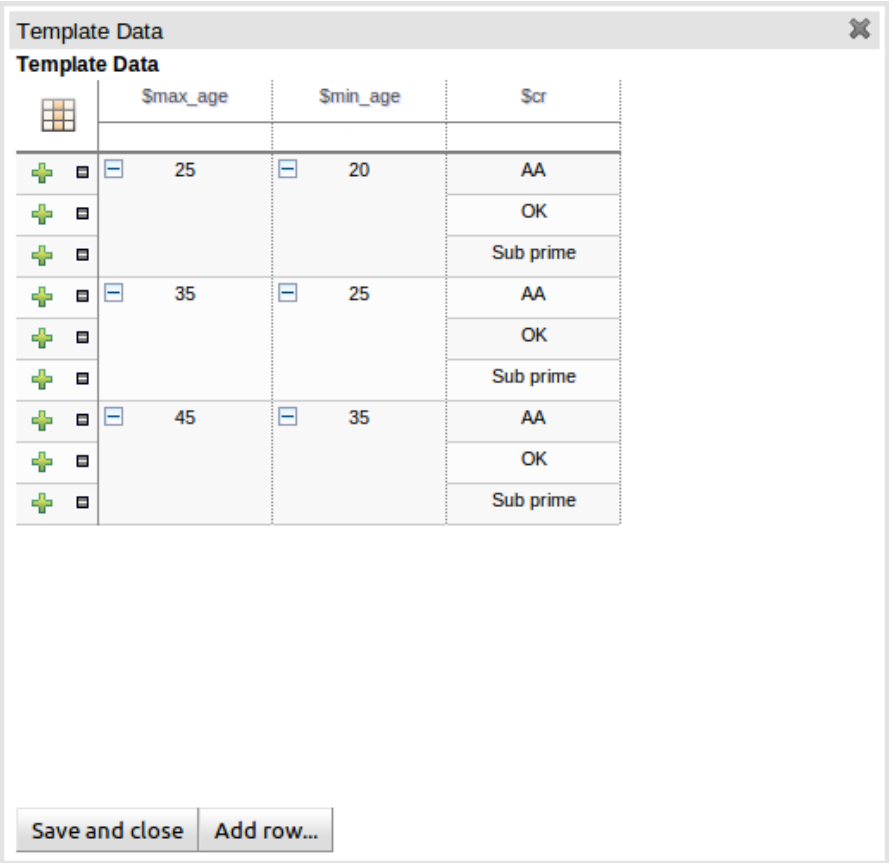


Figura 4.15. Fusión de celdas

4.3.10.1.2. Agrupamiento de celdas

Las celdas que se han fusionado se pueden plegar en una sola fila. El hacer clic en el ícono [+\-] en la parte superior izquierda de una celda fusionada pliega las filas correspondientes en una sola entrada. Las celdas en otras columnas que abarcan las filas plegadas que tienen valores idénticos se muestran sin ningún cambio. Las celdas en otras columnas que abarcan las filas plegadas que tienen valores diferentes se resaltan y presentan el primer valor.

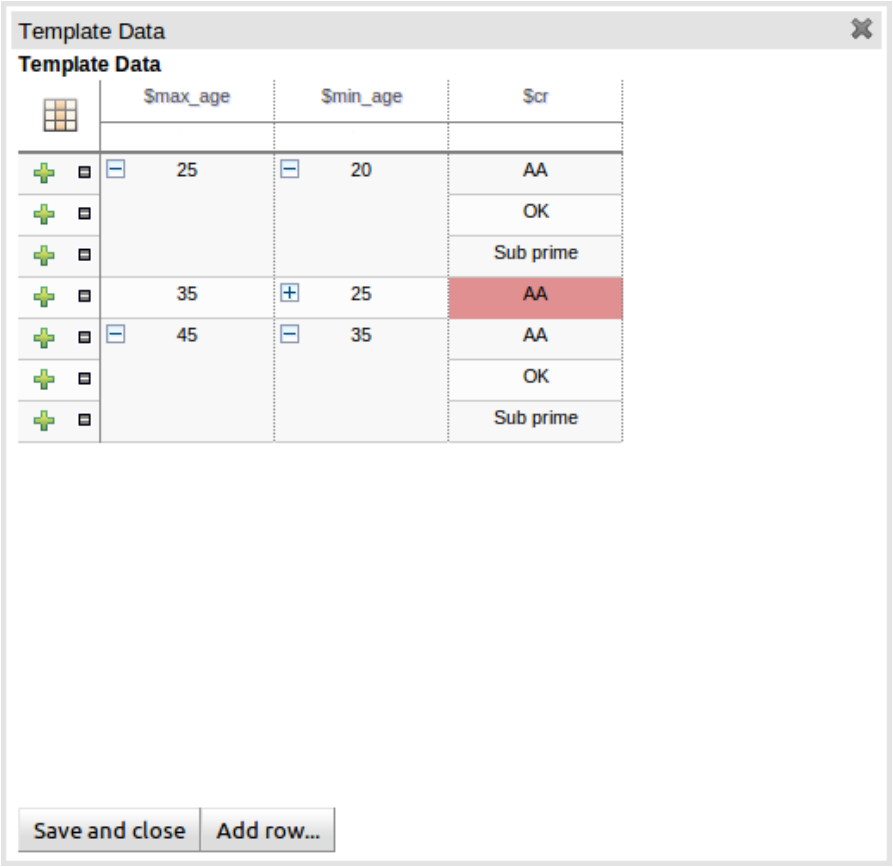


Figura 4.16. Agrupamiento de celdas

Cuando se altera el valor de una celda agrupada entonces también se actualizan los valores de las celdas que se han plegado.

4.3.10.2. DRL generado

Aunque no es necesario los autores de reglas pueden ver el DRL que se generará para una plantilla de reglas y los datos asociados. Esta funcionalidad y su operación no es diferente a la de otros activos. Seleccione la **`Source`** y luego **`View Source`** de la pantalla del editor de activos. Se puede ver el DRL ara todas las reglas.

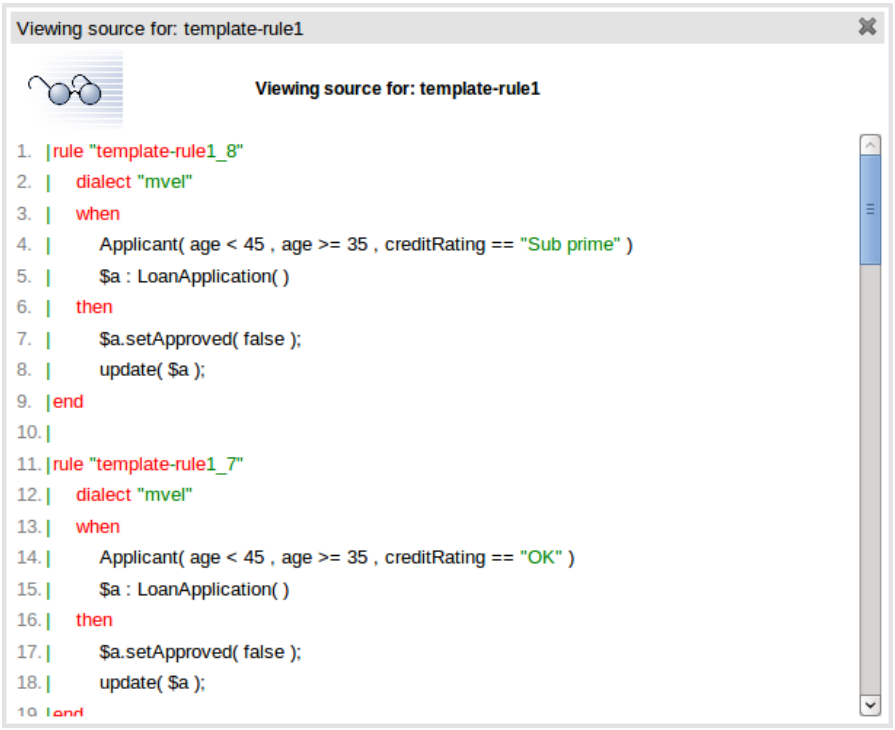

Viewing source for: template-rule1
Viewing source for: template-rule1
1. |rule "template-rule1_8"
2. | dialect "mvel"
3. | when
4. | Applicant(age < 45 , age >= 35 , creditRating == "Sub prime")
5. | $a : LoanApplication( )
6. | then
7. | $a.setApproved(false);
8. | update($a);
9. |end
10. |
11. |rule "template-rule1_7"
12. | dialect "mvel"
13. | when
14. | Applicant(age < 45 , age >= 35 , creditRating == "OK")
15. | $a : LoanApplication( )
16. | then
17. | $a.setApproved(false);
18. | update($a);

Figura 4.17. DRL generado

4.3.11. Flujo de reglas

Flujo de reglas: el flujo de reglas le permite describir visualmente los pasos a tomar así que no todas las reglas se evalúan al mismo tiempo sino que hay un flujo de la lógica. En este capítulo no abordamos los flujos de reglas pero puede utilizar el IDE para dibujar gráficos de flujos de reglas y cargar el archivo .rfm en el BRMS.

De manera similar que las hojas de cálculo puede cargar/descargar archivos de flujos de reglas (el IDE de eclipse tiene un editor gráfico para estos). Los detalles de los flujos de reglas no se discuten aquí.

4.3.12. Reglas técnicas (DRL)

Las reglas técnicas (DRL) se almacenan como texto y se pueden administrar en el BRMS. Un archivo DRL puede contener una o más reglas. Si el archivo contiene solo una regla entonces el paquete importa y no se requieren las declaraciones de reglas. Simplemente puede utilizar "when" y "then" para marcar las secciones de "Condición" y "Acción" respectivamente.

Normalmente utilizaría el entorno de desarrollo integrado para modificar archivos "raw" DRL ya que tiene todas las herramientas avanzadas, ayuda de contenido y funcionalidad de depuración. Sin embargo, hay ocasiones en las que una regla tenga que tratar con algo más bien técnico en un paquete en BRMS. En cualquier paquete típico de reglas generalmente necesita algunas "reglas técnicas". Puede mezclar todos los tipos de reglas claro está.

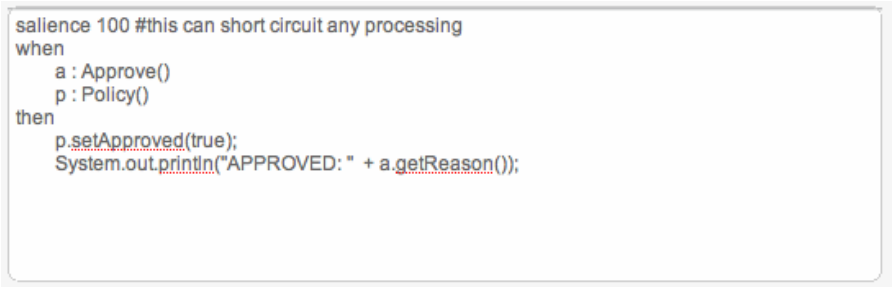


Figura 4.18. Regla técnica DRL

4.3.13. Funciones

Las funciones son otro tipo de activo. No son reglas y solo se deben utilizar cuando sea necesario. El "*Editor de funciones*" es un editor textual.

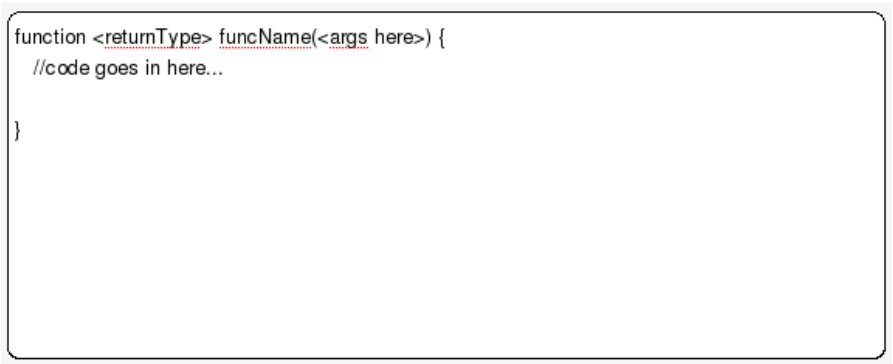


Figura 4.19. Función

4.3.14. Enumeraciones de datos (Configuraciones de la lista desplegable)

Las enumeraciones de datos son un tipo opcional de activo que se puede configurar para proporcionar listas desplegables para el editor dirigido. Se almacenan y se modifican como cualquier otro activo y solo se aplican al paquete al cual pertenecen.

El contenido de una configuración de enumeración es el mapeo de un "Fact.field" a una lista de valores. Estos valores se utilizan para poblar una lista desplegable. La lista puede ser literal o puede utilizar una clase de funcionalidad (la cual pone en la ruta de clase) para cargar las cadenas. Las cadenas contienen un valor que se puede ver en un menú desplegable o un mapeo desde el valor del código (lo cual es lo que se termina utilizando en la regla) y un valor de presentación (vea el ejemplo a continuación que usa el '=').

```
'Board.type' : [ 'Short', 'Long', 'M=Mini', 'Boogie']
'Person.age' : [ '20', '25', '30', '35' ]
```

En la configuración de enumeración anterior la "M" indica un valor que se utilizará en la regla pero "Mini" es lo que se presenta en la interfaz gráfica del usuario.

Obtención de listas de datos de fuentes externas

Es posible tener el código de llamada BRMS, el cual cargará una lista de cadenas. Para lograr esto, necesitará un pedazo de código que retorne un **`java.util.List`** (de cadenas) que esté en la ruta de

clase del BRMS. En lugar de especificar una lista de valores en el BRMS mismo, el códeigo puede retornar la lista de cadenas, (como siempre puede utilizar el signo "=" dentro de las cadenas si quiere utilizar una valore de presentación diferente al valor de la regla). Por ejemplo puede cambiar la línea **`Person.age`** así:

```
'Person.age' : (new com.yourco.DataHelper()).getListOfAges()
```

Esto asume que tiene una clase llamada **`DataHelper`**, la cual tiene un método **`getListOfAges()`**, el cual retorna una lista de cadenas (y se encuentra en la ruta de clase). Por supuesto que puede mezclar estas enumeraciones "dinámicas" con listas fijas. Por ejemplo, podría cargarlas desde una base de datos usando JDBC. Las enumeraciones de datos se cargan la primera vez que utilice el editor dirigido en una sesión. Si tiene alguna sesión abierta del editor dirigido será necesario que la cierre y vuelva a abrir la regla para poder ver el cambio. Para verificar que la enumeración se ha cargado, vaya a la pantalla de configuración del paquete. Puede "guardar y validar" el paquete; esto lo chequeará y brindará cualquier comentario sobre los errores.

4.3.15. Conceptos avanzados de enumeración

Hay otras pocas tareas avanzadas que puede llevar acabo con enumeraciones de datos.

Listas desplegables dependientes de valores de campos

Imagínese un modelo simple de hechos, en el cual tiene una clase llamada Vehicle -vehículo que tiene dos campos: "**`engineType`**" y "**`fuelType`**." Quiere tener opciones para "**`engineType`**" de "**`Petrol`**" o "**`Diesel`**." Obviamente la opción del combustible debe ser dependiente del tipo de máquina (para gasolina tenemos ULP y PULP y para Diesel tenemos BIO y **`NORMAL`**). Podemos expresar esta dependencia en una enumeración así:

```
'Vehicle.engineType' : ['Petrol', 'Diesel']
'Vehicle.fuelType[engineType=Petrol]' : ['ULP', 'PULP' ]
'Vehicle.fuelType[engineType=Diesel]' : ['BIO', 'NORMAL' ]
```

Esto muestra como es posible realizar selecciones dependientes en otros valores de campos. Note que una vez seleccione el **`engineType`** se determina la "lista seleccionada" para el **`fuelType`**.

Carfa de enumeraciones programáticamente

En algunos casos se querrá cargar los datos de enumeración desde una fuente de datos externa (como una base de datos relacional). Para lograr esto, puede implementar una clase que retorne un mapa. La clave del mapa es una cadena (la cual es el nombre **`Fact.field`** que vimos anteriormente). El valor es una **`java.util.List`** de cadenas.

```
public class SampleDataSource2 {

  public Map<String>, List<String>> loadData() {
    Map data = new HashMap();

    List d = new ArrayList();
    d.add("value1");
    d.add("value2");
    data.put("Fact.field", d);

    return data;
 }

}
```

En la enumeración en el BRMS ponga lo siguiente:

```
=(new SampleDataSource2()).loadData()
```

El signo "=" le dice que cargue los datos ejecutando su código.

Enumeraciones avanzadas

En los casos que vimos anteriormente los valores en las listas se calculan "por adelantado". Esto es muy adecuado para cantidades de datos relativamente estáticos o pequeños. Sin embargo, imagínese un escenario en donde tiene listas de países y cada país tiene una lista de estados, cada estado tiene una lista de localidades, cada localidad tiene una lista de calles, etc. Podrá ver que esto puede llegar a una gran cantidad de datos. Dicha cantidad no se podría cargar. En lugar, las listas se podrían cargar dependiendo del país que se selecciona.

Esta situación se puede abordar de la siguiente manera:

```
'Fact.field[dependentField1, dependentField2]' : '(new
com.yourco.DataHelper()).getListOfAges("@{dependentField1}", "@{dependentField2}")'
```

Observe que solo se han especificado los campos que se necesitan. También a la derecha del signo ":" hay comillas alrededor de la expresión. Esta expresión se evaluará solo cuando se necesite. Cuando lo hace substituirá los valores de los campos especificados. Esto significa que puede utilizar los valores de los campos de la interfaz gráfica del usuario para dirigir una petición a la base de datos y profundizar en los datos, etc. Cuando se carga la lista desplegable de reglas, la lista se refrescará con base en esos campos. "**`depenentField1`**" y "**`dependentField2`**" son nombres de campos en el tipo "Fact". Se utilizan para calcular la lista de valores que se van a ver en una lista desplegable como valores para el "campo".

4.4. Administración del estatus

Cada activo y paquete en el BRMS tiene una *etiqueta de estatus* establecida. Los valores para la configuración de etiquetas de estatus se encuentran en la sección "Administración" del BRMS. Aquí puede agregar sus propios nombres de estatus. De manera similar a las categorías, los estatus NO afectan la ejecución de ninguna manera. Sólo proporcionan información.

Nota

De manera opuesta a las categorías, los activos solo pueden tener un estatus a la vez.

El uso de los estatus es completamente opcional. Puede utilizar los estatus o las categorías para administrar el ciclo de vida de los activos.

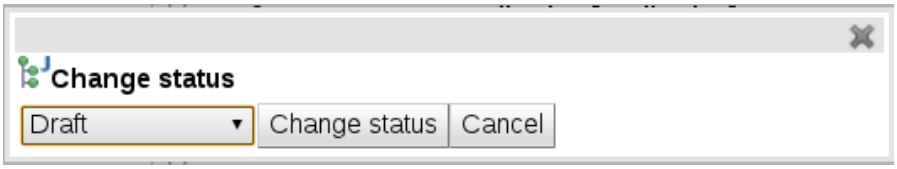


Figura 4.20. Estatus de los activos

El diagrama anterior presenta un cambio en el estatus de un activo individual. El cambio tiene efecto de manera inmediata y no se necesitar guardar por separado.

También puede cambiar el estatus de un paquete completo. Esto establece la etiqueta del estatus en el paquete mismo y en todos los activos que pertenecen al mismo valor.

4.5. Administración de paquetes

En el BRMS tiene una "*base de conocimiento*," la cual contendrá uno o más "*paquetes de conocimiento*." En la interfaz del usuario, los paquetes de conocimeinto con frecuencia se conocen simplemente como "*paquetes*."

Aviso

El configurar paquetes de conocimeinto usualmente es una tarea que sólamente se lleva a cabo una sola vez y que la realiza alguien con experiencia en desarrollo de reglas y modelos. Muy pocas personas necesitarán configurar los paquetes de conocimiento y una vez que estén listos se pueden copiar sucesivamente si es necesario. La configuración de paquetes es definitivamente una tarea técnica que requiere experiencia con el desarrollo de reglas.

Todos los activos viven en "paquetes" en el BRMS. Un paquete es como un sub-directorio (y también sirve como un "espacio de nombre"). Es el equivalente a una carpeta de inicio, en la cual viven los activos de reglas. Las reglas en particular necesitan saber la naturaleza del modelo de hechos y el espacio de nombres.

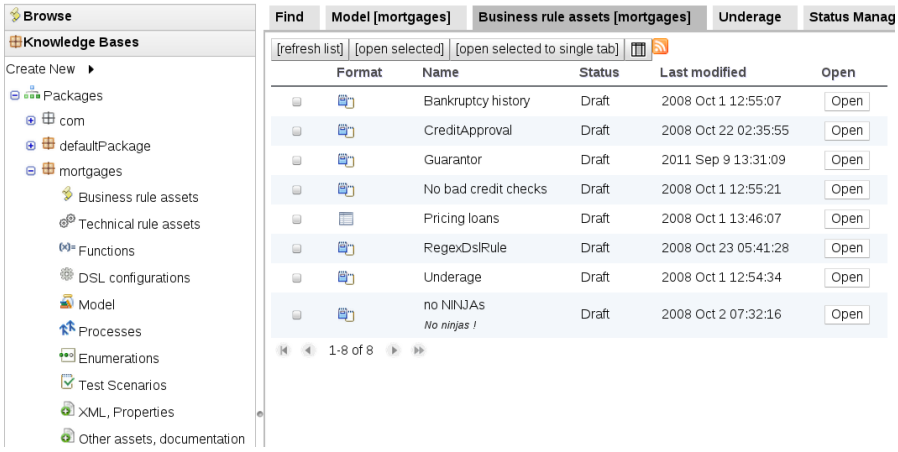


Figura 4.21. El explorador de paquetes

La figura anterior muestra el *explorador de paquetes*. Al hacer clic en un tipo de activos mostrará una lista de coincidencias (para los paquetes con miles de reglas y puede que tome varios segundos para que se presente la lista, de ahí viene la importancia del uso de categorías para una navegación rápida).

Para resumir, aunque las reglas (y activos en general) peuden aparecer en cualquier número de categorías, sólo viven en un paquete. Si considera el BRMS como un sistema de archivos entonces cada paquete es un directorio y los activos viven en ese directorio como una lista de archivos. Cuando crea una toma de pantalla de una implementación de un paquete, está efectivamente copiando todos los activos de ese "directorio" en otro "directorio" especial.

La funcionalidad de administración de paquetes le permite ver una lista de paquetes de conocimiento y puede "expandirlos" para ver listas de cada "tipo" de activo, (hay muchos activos así que algunos de ellos se agrupan juntos):

Los tipos de activos:

- Activos empresariales: este muestra una lista de todos los tipos de "reglas empresariales", incluyendo tablas de decisiones, reglas empresariales, etc.
- Activos técnicos: esta es una lista de cosas que se consideran como técnicas (tal como reglas DRL, enumeraciones de datos y flujos de reglas).
- Funciones: en BRMS también puede tener funciones definidas, (estos es opcional).
- DSL: del inglés Domain Specific Languages - lenguajes específicos de dominios - también se pueden almacenar como activos. Si existen (generalmente solo habrá uno) entonces se utilizarán junto con los GUIs del editor apropiado.
- Modelo: un paquete requiere por lo menos un modelo. Este es para las reglas.

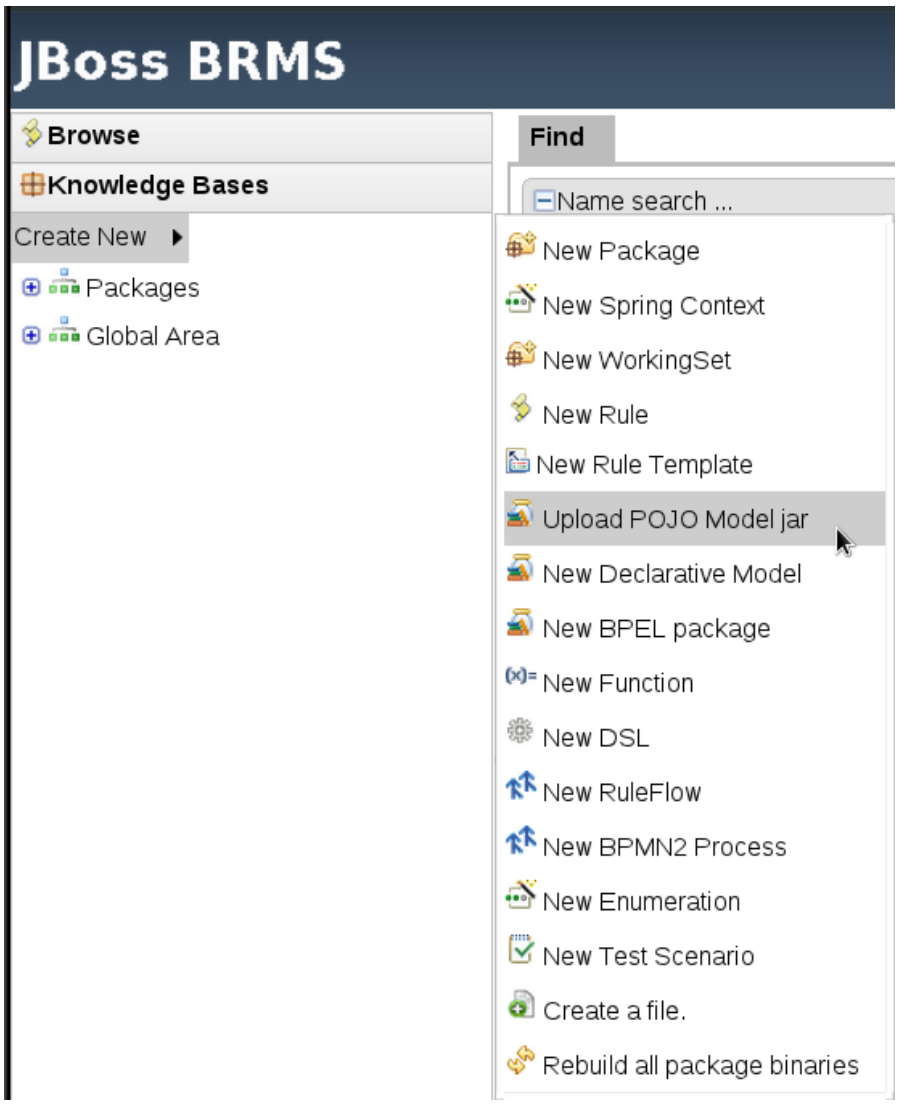


Figura 4.22. Creación de nuevos activos

Puede crear nuevas reglas o nuevos activos usando el explorador de paquetes. Algunos activos sólo se pueden crear por medio del explorador de paquetes. Figura 4.22, “Creación de nuevos activos” muestra los íconos que inicial los "asistentes" para este propósito.

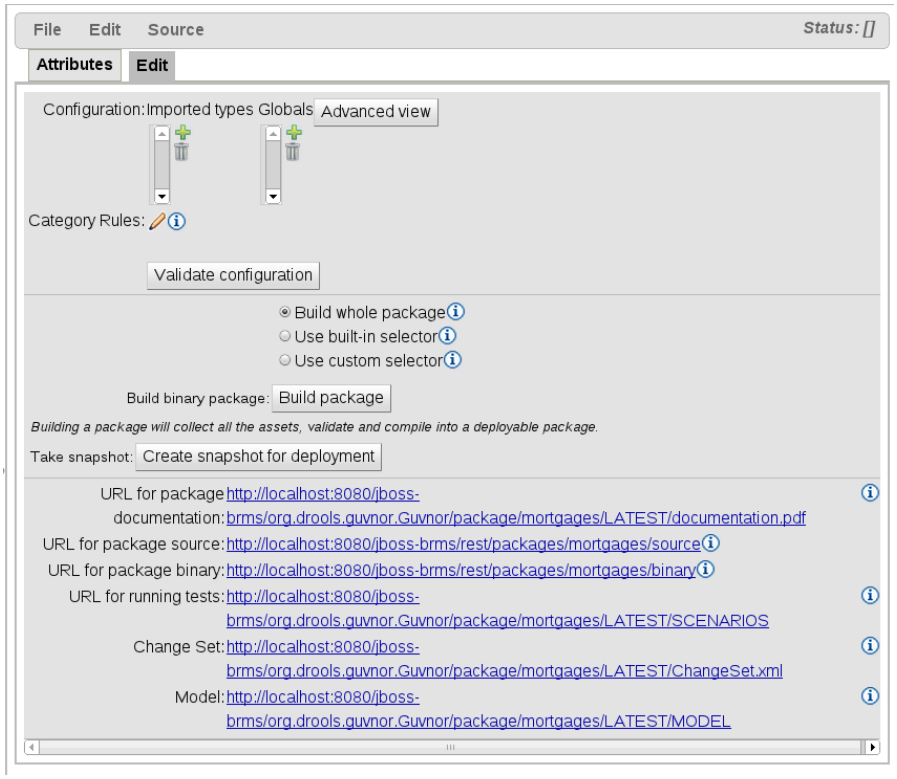


Figura 4.23. Configuración de paquetes

Una de las cosas más importantes que tiene que hacer es configurar paquetes. Esta tarea implica dos aspectos de la importación de clases que las reglas utilizan y luego definir variables globales. Una vez que realiza un cambio es necesario guardarlo. En ese momento, el paquete está configurado y está listo para construir. Por ejemplo, puede agregar un modelo, el cual tiene una clase llamada **`com.something.Hello`**. Luego agregaría **`import com.something.Hello`** en su configuración de paquete y guardaría el cambio.

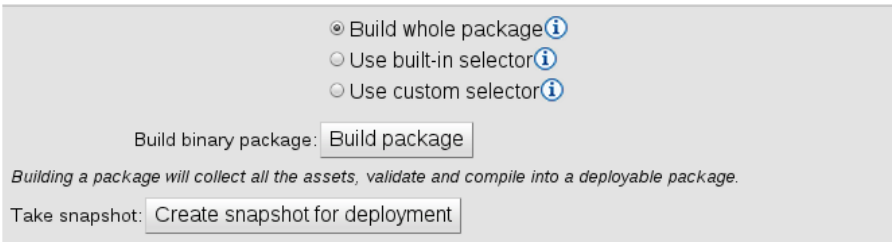

Build whole package
Use built-in selector
Use custom selector
Build binary package:
Build package
Building a package will collect all the assets, validate and compile into a deployable package.
Take snapshot:
Create snapshot for deployment

Figura 4.24. Construcción de paquetes

Después de importar las clases del paquete y de definir sus variables globales, se construye el paquete. Entre más activos tenga un paquete entonces tomará más tiempo en construírse. Si la construcción tiene éxito entonces tendrá la opción de crear una "toma de pantalla" para la implementación. En ese punto también puede ver el "drl" que se genera cuando se construye el paquete.

4.5.1. Importación de paquetes drl

También es posible crear un paquete importando un archivo "drl" ya existente. Cuando decide crear un nuevo paquete tiene la opción de cargar un archivo **`.drl`**. BRMS tratará de comprender ese drl y creará automáticamente un paquete por usted. Las reglas en este se almacenarán como activos individuales (aunque todavía serán contenido de texto drl). Note que para contruir el paquete necesitará cargar un

modelo apropiado (como un **.jar**) frente al cual validar. Este es un paso separado.

4.6. Administración de versiones

Tanto paquetes de conocimiento como activos individuales tienen *versiones* en el BRMS pero el mecanismo es un poco diferente para cada uno. Los activos individuales se guardan como una versión de un archivo en un sistema de control de fuentes tal como **Subversion**. Sin embargo, los paquetes de activos tienen versiones "a demanda" realizando una toma de pantalla. Esta toma de pantalla es la que se utiliza para la implementación. La siguiente sección aborda la administración de la implementación y las tomas de pantalla con más detalles.

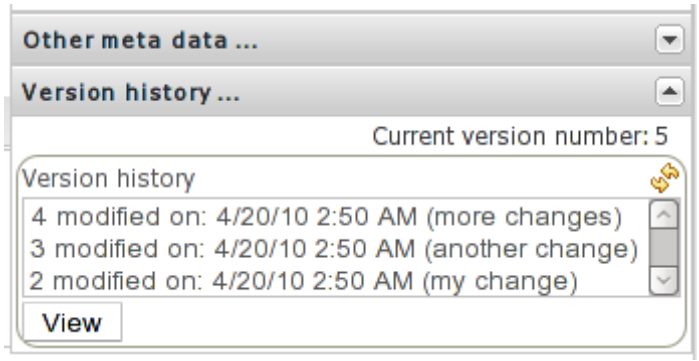

Other meta data ...
Version history ...
Current version number: 5
Version history
4 modified on: 4/20/10 2:50 AM (more changes)
3 modified on: 4/20/10 2:50 AM (another change)
2 modified on: 4/20/10 2:50 AM (my change)
View

Figura 4.25. Versiones de activos

Cada vez que realiza un cambio a un activo se crea un nuevo objeto en el historial de versiones. Esto le brinda una función ilimitada para "deshacer". Puede mirar la historia de un activo individual, (como se puede ver en la lista anterior) y luego verlo y reestablecerlo desde ese momento.

4.7. Administración de implementación

Las URLs son una parte central del proceso que rodea la manera en que se proporcionan los paquetes de conocimiento. BRMS proporciona paquetes de conocimiento por medio de URLs para que el agente de conocimiento los utilice y los descargue. Estas URLs tienen la forma: http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/<packageName>/<packageVersion>

<packageName> es el nombre que le dio al paquete. <packageVersion> es el nombre o una toma de pantalla o "**LATEST**" (si es "**LATEST**," entonces será la última versión construída del paquete principal y no una toma de pantalla). Puede utilizarlas en el agente o puede pegarlas en su navegador y las descargará como archivos.

Consulte Sección 8.1, “El agente de conocimiento” para obtener mayores detalles sobre cómo utilizar estas URLs y las descargas binarias en su aplicación y la manera en que se pueden actualizar las reglas "sobre la marcha".

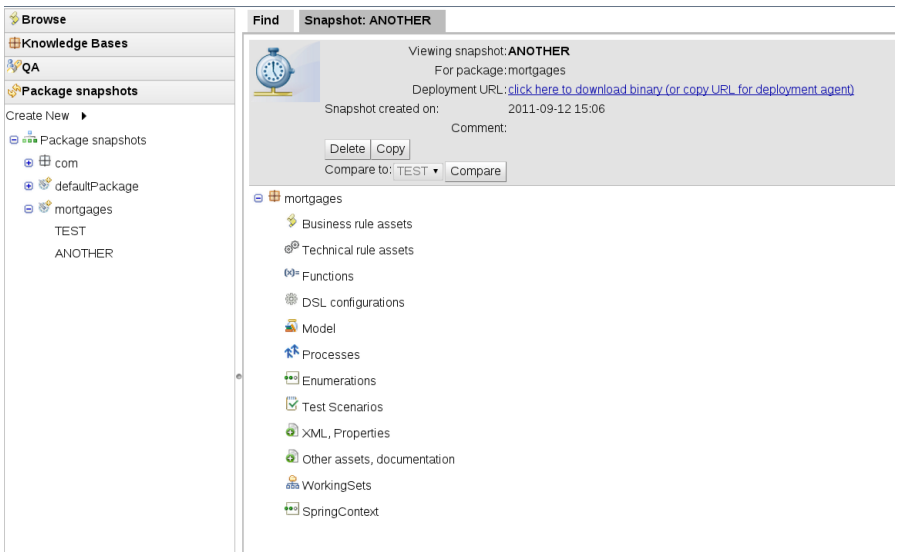


Figura 4.26. Tomas de pantallas de implementaciones

La imagen anterior muestra la vista *Deployment Snapshots*. En la izquierda hay una lista de paquetes de conocimiento. Al hacer clic en un paquete en especifico en esa lista podrá ver todas las tomas de pantalla para este (si hay alguna). Desde allí puede copiar, borrar o ver una toma de pantalla. Cada toma de pantalla está disponible para descargarse o para accederla por medio de una URL. Una vez descargada se pueden implementar.

4.8. Navegación del repositorio y ubicación de reglas

Las dos maneras principales de ver el repositorio es utilizar la categorización dirigida por el usuario (también conocida como etiquetación) y la vista *Package Explorer*.

La vista de categorías le brinda una manera de navigar sus reglas que tengan sentido para su organización.

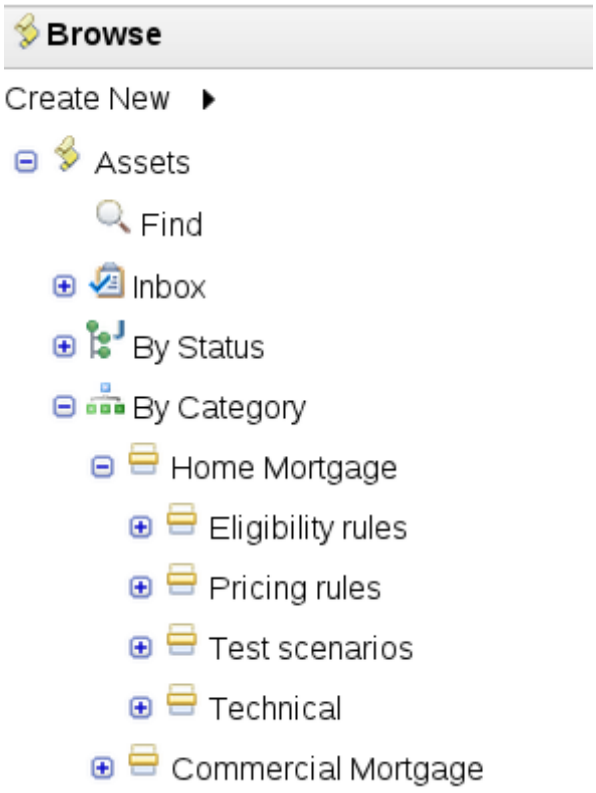


Figura 4.27. Vista de categoría

El diagrama anterior muestra la funcionalidad de categoría en acción. Si es posible limite el número de reglas en cada categoría a máximo unas pocas docenas.

La vista opcional usa el explorador de paquetes. Esta vista muestra los activos de una manera que refleja cómo están de hecho almacenados en la base de datos. También separa reglas en paquetes de conocimiento y su tipo o formato.

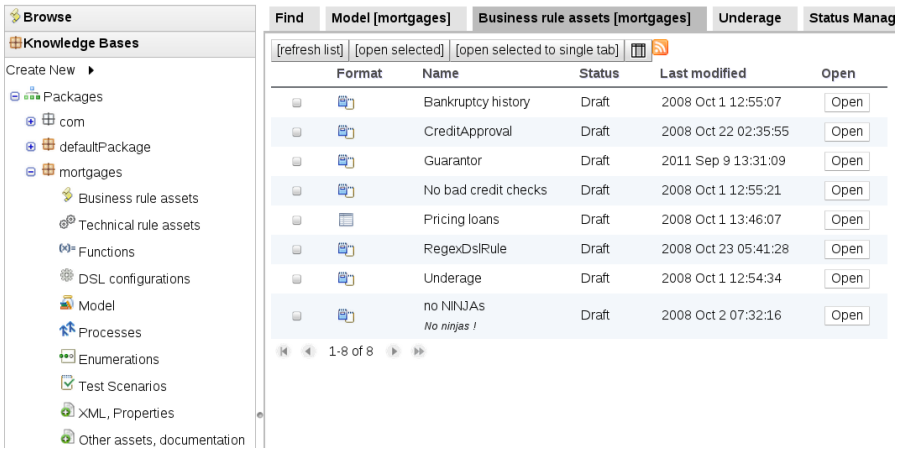


Figura 4.28. Vista de paquete

Capítulo 5. El modelo de hechos (el modelo de objetos)

Se necesita un *modelo de hechos* para dirigir las reglas de una aplicación basada en reglas. El modelo de hechos usualmente coincide con el *modelo de dominio* de la aplicación pero en general se debe separar de este. Esto hace más fácil el administrar las reglas con el pasar del tiempo.

Hay dos maneras de definir un modelo de hechos:

- Cargue un archivo **JAR** que contenga las clases Java que la aplicación y las reglas utilicen.
- Declare un modelo dentro de BRMS que se pueda exportar como una **KnowledgeBase** y se utilice dentro de su código Java.

5.1. El área global

Cuando se crean modelos de hechos se pueden importar en el paquete específico con el que se utilizarán o se pueden importar en el **`Global Area`**. Es importante observar que los activos que se encuentran en el **`Global Area`** no están disponibles globalmente se tienen que importar en los paquetes que los utilizarán. El **`Global Area`** se debe utilizar como un lugar de almacenamiento para los activos que todavía no se están utilizando o como punto central para los activos utilizados en múltiples paquetes.

NOTA

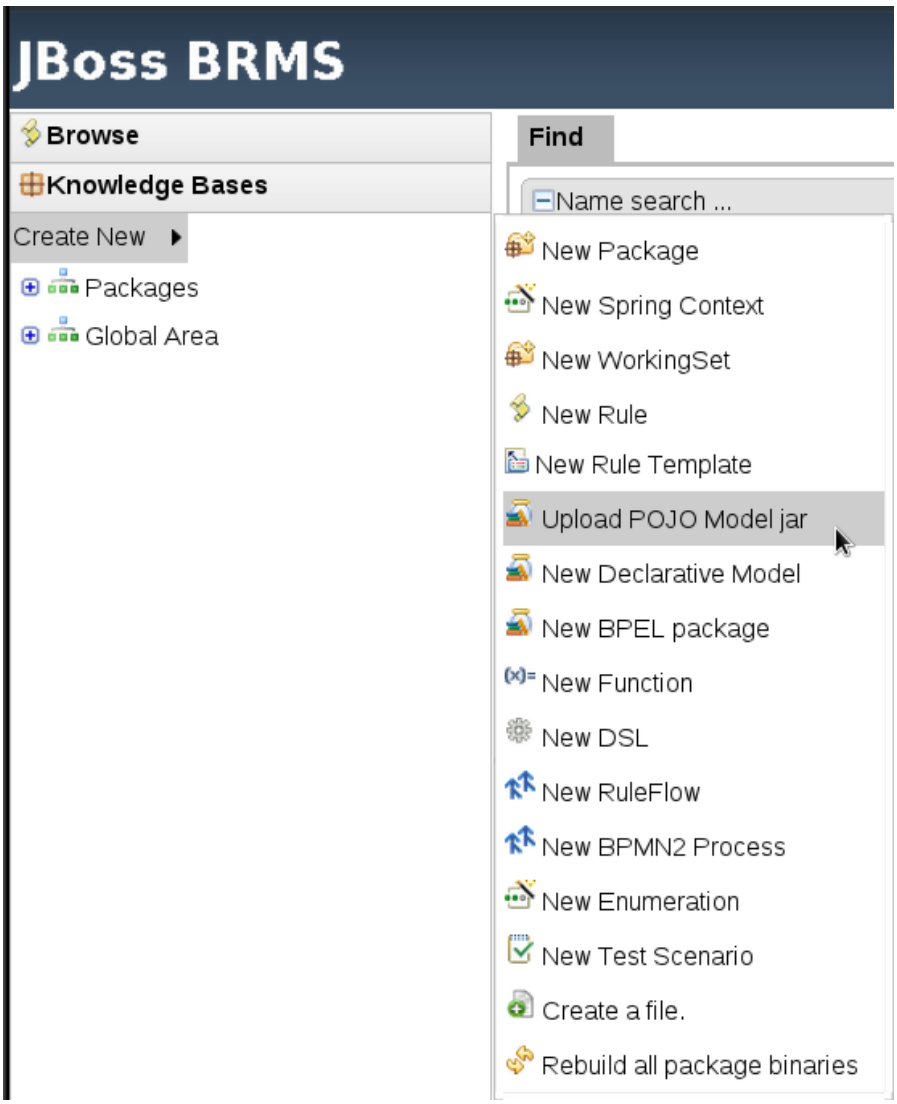
Tenga cuidado al modificar los activos que se encuentren en ambos paquetes y en el **`Global Area`**. Será necesario importar de nuevo los activos modificados en el **`Global Area`** en los paquetes que los utilizan y borrar los activos importados anteriormente.

5.2. Modelo JAR

Procedimiento 5.1. Creación de un modelo Jar

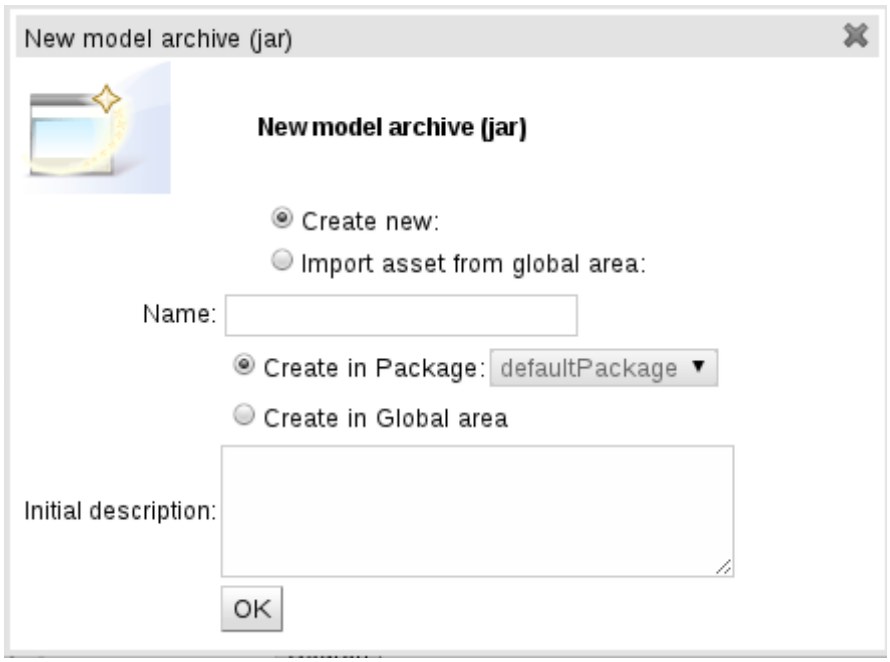
1. **Abra el menú `New model archive (jar)`**

   Del menú **`Knowledge Bases`** seleccione **`Create New`** y luego seleccione **`Upload POJO Model JAR`**.



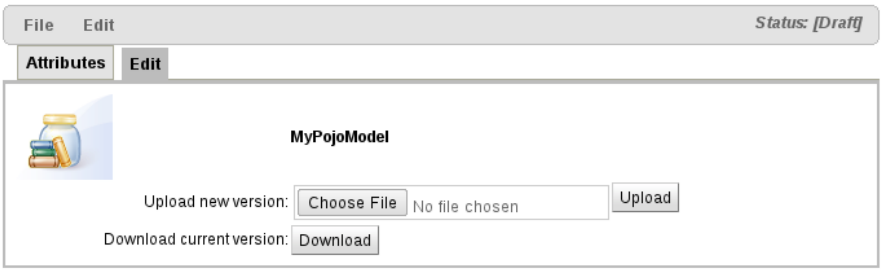
2. **Creación de un activo modelo Jar**

   Introduzca el nombre del modelo Jar, la categoría y una descripción. Seleccione en que paquete crear el modelo o especifique que se debe agregar al **`Global Area`**. Haga clic en **`OK`** cuando todos los detalles se han introducido.



3. **Cargue la JAR en el activo**

Cargue la JAR que contiene el modelo definido como paquetes y clases Java y en un archivo Java JAR normal.



5.3. Modelo declarativo

El utilizar un modelo declarativo tiene los siguientes beneficios:

- Refuerza que el modelo pertenece a la base de conocimiento, note la aplicación.
- El modelo puede tener un ciclo de vida separado de las aplicaciones.
- Los tipos Java se pueden enriquecer con anotaciones especificas de reglas.
- Los archivos JAR se deben mantener sincronizados entre las reglas y las aplicaciones que las utilizan, sin embargo, no es necesario mantener sincronizado un modelo declarativo.

Los modelos declarativos pueden ser:

- Una definición autónoma de todo el modelo de hechos que se utiliza dentro de sus reglas.
- Una definición de hechos complementaria para soportar un modelo Java POJO.

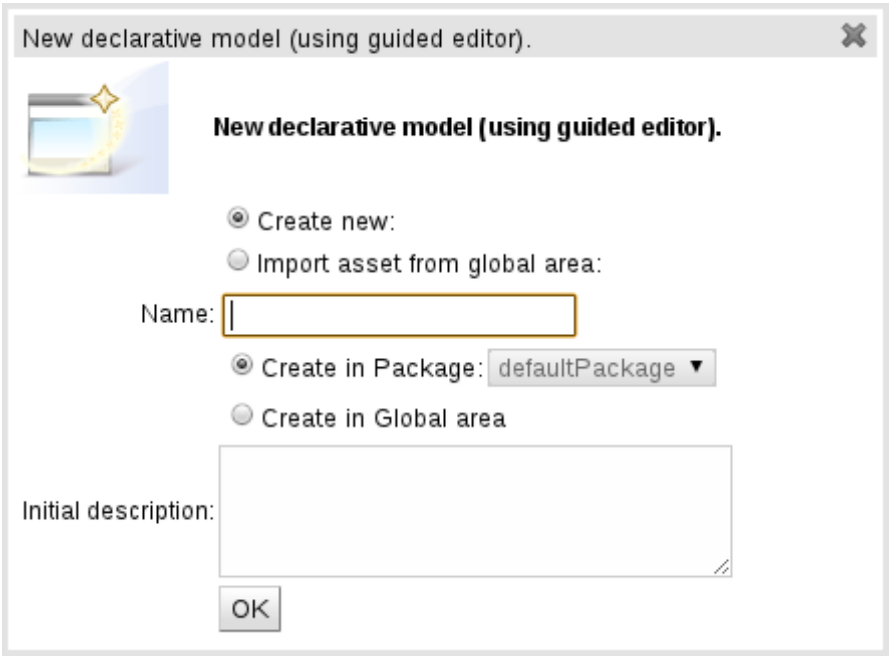
Procedimiento 5.2. Creación de un modelo declarativo

1. **Abra el menú `New Declarative Model`**

   Del menú **`Knowledge Bases`** seleccione **`Create New`** y luego seleccione **`New Declarative Model`**.

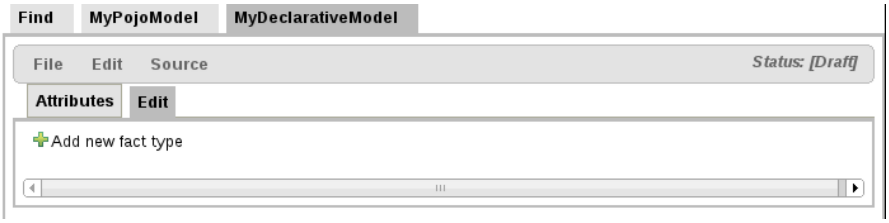
2. **Creación de un nuevo modelo declarativo**

   Introduzca un nombre para el nuevo modelo. Seleccione el paquete en donde crear el modelo o especifique que se debe agregar al **`Global Area`**. Haga clic en **`OK`** cuando se hayan introducido todos los detalles.



3. **Defina el modelo**

   Haga clic en **`Add new fact type`** e introduzca el nombre del hecho en el campo **`name`** del menú emergente.

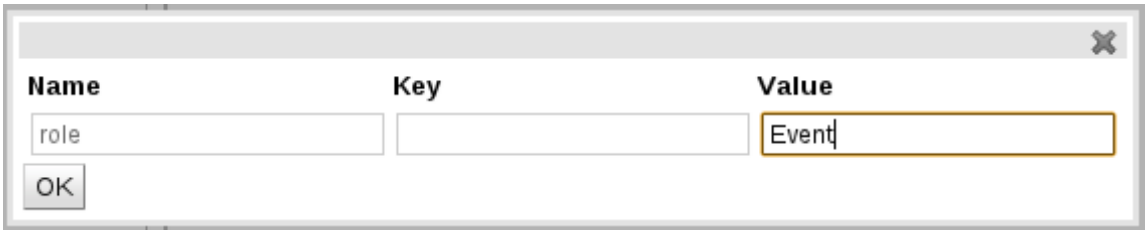


4. **Agregar campos de hechos**

   Cree campos de hechos seleccionando el botón **`Add field`** e introduciendo la información en el menú emergente.

5. **Agregar anotaciones**

   Cree anotaciones de hechos seleccionando el botón **`Add annotation`**. Los campos **`Name`** y **`Value`** de las anotaciones son obligatorios pero el campo **`Key`** es opcional. Si no se especifica un valor **`Key`** entonces se asigna un valor predeterminado ***`value`***.



5.3.1. Consumo de un modelo declarativo de Java

Los tipos declarados se generan en el momento de la compilación de la base de conocimiento y como tal la aplicación sólo tendrá acceso a estos en tiempo de ejecución de la aplicación. Por lo tanto, estas clases no están disponibles para referencia directa de la aplicación.

Los tipos declarativos se pueden usar como objetos de hechos normales pero la manera de crearlos es diferente (ya que no se encuentran en la ruta de clase de las aplicaciones). Para crear estos objetos están disponibles de la instancia KnowledgeBase.

Ejemplo 5.1. Manejo de tipos de hechos declarados por medio de la API

```
// obtienen una referencia a una base de conocimiento con un tipo declarado:
KnowledgeBase kbase = ...

// obtiene el FactType declarado
FactType personType = kbase.getFactType( "org.drools.examples",
                                         "Person" );

// maneja el tipo como sea necesario:
// create instances:
Object bob = personType.newInstance();

// establece los valores de los atributoss
personType.set( bob,
                "name",
                "Bob" );
personType.set( bob,
                "age",
                42 );

// introduce el hecho en una session
StatefulKnowledgeSession ksession = ...
ksession.insert( bob );
ksession.fireAllRules();

// lee los atributos
String name = personType.get( bob, "name" );
int age = personType.get( bob, "age" );
```

Nota

El espacio de nombre del tipo declarado es el espacio de nombre del paquete en donde se declaró (es decir **`org.drools.examples`** en el ejemplo anterior).

Capítulo 6. grupos de trabajo

NOTA

Los grupos de trabajo son una funcionalidad experimental disponible en BRMS 5.2.

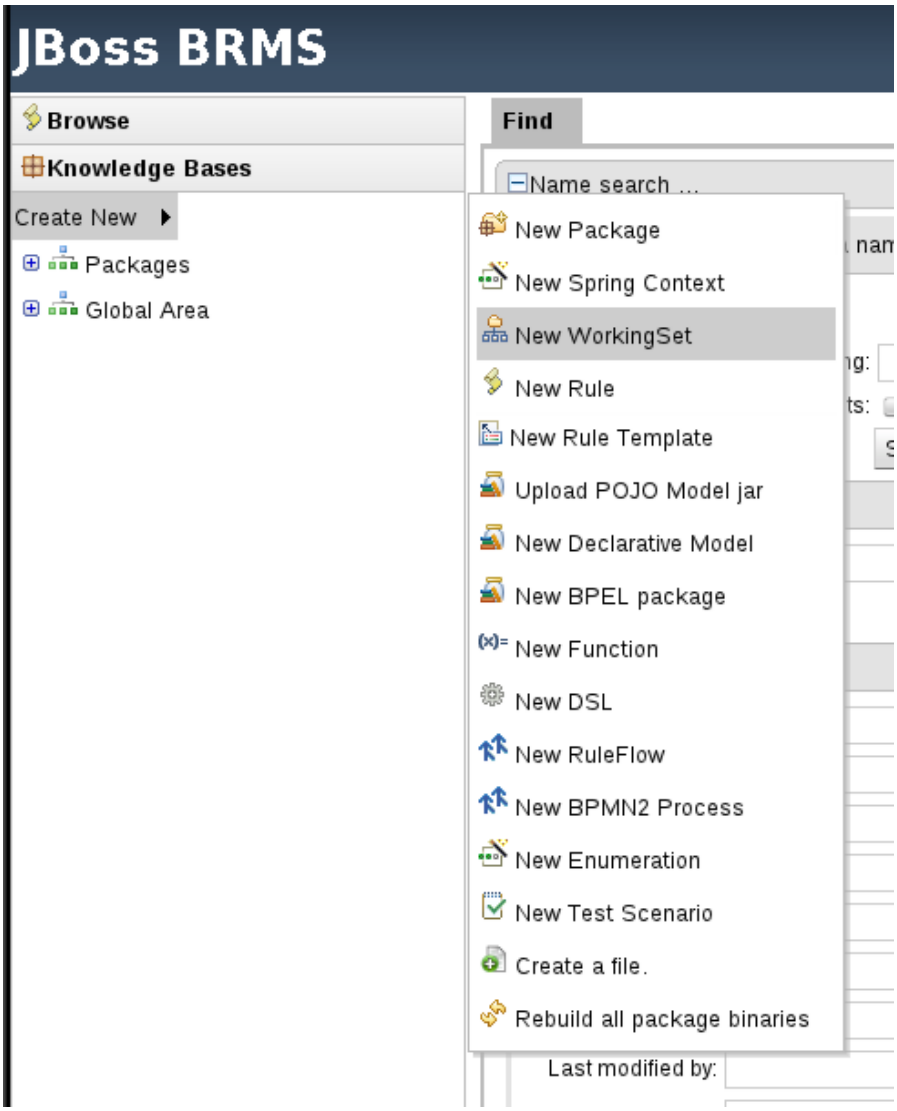
Los grupos de trabajo agrupan los hechos y definen restricciones comunes para ellos. Los grupos de trabajo también hacen posible el limitar las reglas que son visibles en el editor dirigido al escribir reglas.

Los grupos de trabajo se deben activar de manera manual desde el editor dirgido.

Procedimiento 6.1. Creación de un nuevo grupo de trabajo

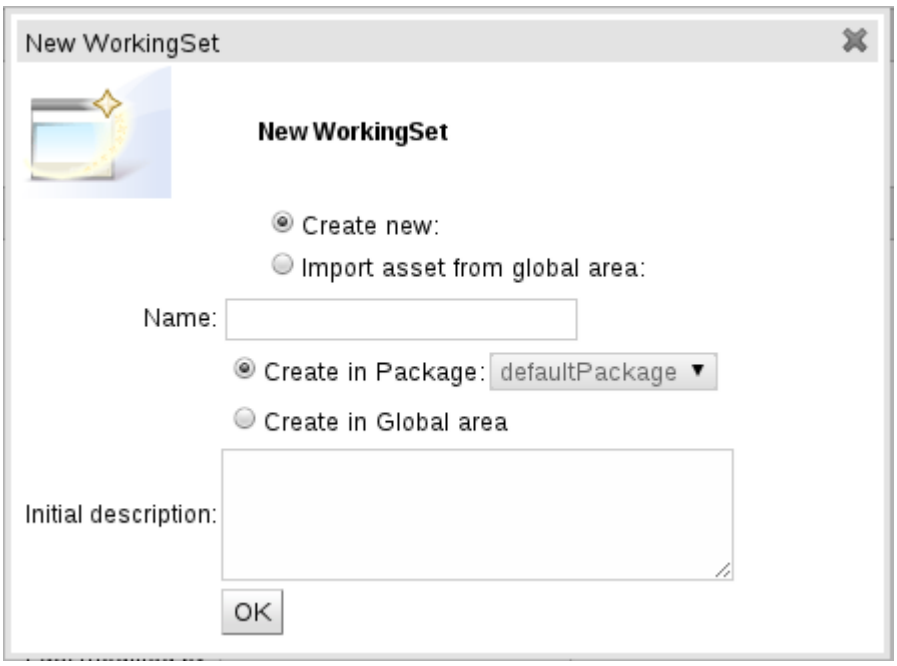
1. **Abra la ventana `New WorkingSet`**

   Del menú **`Knowledge Bases`** seleccione **`Create New`** y luego seleccione **`New WorkingSet`**.



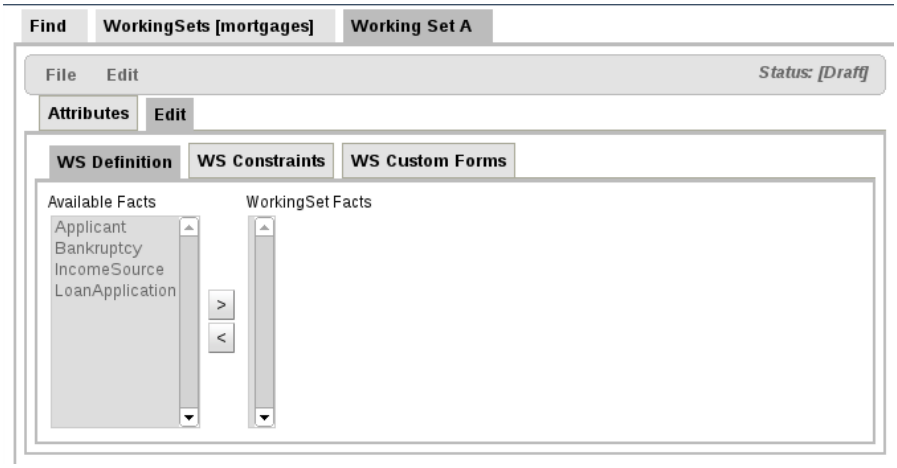
2. **Creación de un nuevo grupo de trabajo**

   Introduzca un nombre y una descripción para el grupo de trabajo. Seleccione el paquete en donde crear el modelo o especifique que se debe agregar al **`Global Area`**. Haga clic en **`OK`** cuando se hayan introducido todos los detalles.



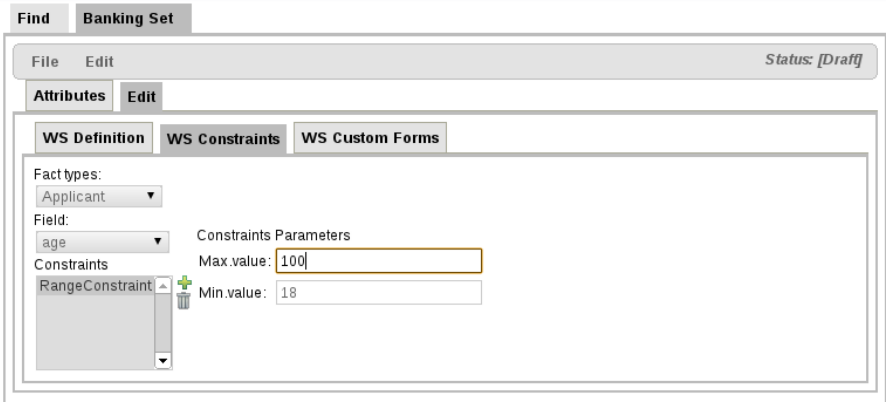
3. **Agregar tipos de hechos al grupo de trabajo**

   Agregue los tipos de hechos al grupo de trabajo moviéndolos de la lista en la izquierda a la lista de la derecha. Los tipos de hechos en la derecha estarán disponibles para el grupo de trabajo.



4. **Agregar restricciones a los tipos de hechos en el grupo de trabajo**

   Seleccione el **`Fact type`** del menú desplegable, seleccione el valor **`Field`** y agregue la restricción requerida.



6.1. Verificación de restricciones de campos

Las restricciones de campos se pueden verificar de dos maneras:

- Validación a demanda.
- Validación en tiempo real.

La validación a demanda se realiza seleccionando **`verify`** de la barra de herramientas del editor dirigido. Se ejecuta un reporte de verificación con los resultados de verificación.

La validación en tiempo real chequea posibles violaciones de restricciones de campos en tiempo real y marca las líneas en donde ocurrieron violaciones.

Para habilitar la verificación en tiempo real seleccione **`Administration`**, luego **`Rules Verification`** y después seleccione la casilla **`Enable`**.

NOTA

Esta es una funcionalidad experimental y está inhabilitada por defecto.

Capítulo 7. La perspectiva del usuario empresarial

Para los usuarios empresariales los formatos de reglas más apropiados a utilizarse son los derivados al utilizar el editor dirigido, las tablas de decisión y las reglas DSL. También puede utilizar algunas expresiones DSL en el editor dirigido (de manera que proporcione "formas" en las que se pueden introducir valores).

Puede utilizar categorías para aislar reglas y activos de los usuarios no-técnicos. Sólo aparecerán los activos que se han asignado a una categoría en la vista "Categorias".

Un desarrollador o persona técnica tendrá que realizar la configuración inicial de la plataforma BRMS. Esta persona establecerá los fundamentos para todas las reglas. También puede crear "plantillas", las cuales son reglas que se pueden copiar (usualmente se encuentran en un paquete "ficticio" bajo la categoría de "plantilla"). Puede llegar a ser útil el utilizar plantillas.

Los usuarios no-técnicos tampoco deben llevar a cabo la implementación. Como lo mencionamos anteriormente en este *manual*, las implementaciones se llevan acabo por medio del sistema "paquetes").

Capítulo 8. Integración de reglas con sus aplicaciones

Hasta este momento el manual ha abordado la administración de reglas. Ahora aprenderemos sobre su uso en su aplicación. Esta sección aborda el uso del componente de implementación del agente de conocimiento que automatiza la mayor parte de este proceso por usted.

8.1. El agente de conocimiento

Importante

El archivo **`README_DEPENDENCIES.txt`** que se incluye en **`jboss-brms-engine.zip`** contiene detalles específicos para las dependencias de cada componente.

El agente de conocimiento es un componente que está incluído en la API de JBoss Rules 5.0. No se necesitan componentes adicional para usar el agente de conocimiento. Si esta utilizando la plataforma JBoss Enterprise BRMS, la aplicación sólo necesita incluír las dependencias **`drools-core`** en su ruta de clase, es decir, sólo las JARs **`drools`** y **`mvel`**. No hay otras dependencias específicas de las reglas.

También hay una tarea ant drools-ant de manera que pueda construir reglas como parte de un script ant y generar archivos .pkg. El escenario más común para esto en cuando las reglas se modifican en un IDE tal como JBoss Developer Studio sin usar la interfaz de usuario de BRMS.

Una vez que haya construído sus reglas en un paquete en la plataforma BRMS (o desde la tarea ant), está listo para utilizar el agente en su aplicación destino.

El siguiente ejemplo construye un agente que construirá una nueva KnowledgeBase desde los archivos especificados en la cadena de la ruta. Sondeará esos archivos cada 60 segundos, el cual es el valor predeterminado, para ver si se han actualizado. Si se encuentran nuevos archivos entonces construirá una nueva KnowledgeBase. Si el grupo de cambios especifica un recurso que sea un directorio entonces su contenido se escaneará en busca de cambios.

```
KnowledgeAgent kagent = KnowledgeAgentFactory.newKnowledgeAgent( "MyAgent" );
kagent.applyChangeSet( ResourceFactory.newUrlResource( url ) );
KnowledgeBase kbase = kagent.getKnowledgeBase();
```

El KnowledgeAgent puede aceptar una configuración que permite cambiar algunos valores predeterminados. Una propiedad de ejemplo es "drools.agent.scanDirectories", por defecto se escanean los directorios especificados en busca de nuevas adiciones pero es posible inhabilitar esto.

```
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();

KnowledgeAgentConfiguration kaconf =
KnowledgeAgentFactory.newKnowledgeAgentConfiguration();
kaconf.setProperty( "drools.agent.scanDirectories", "false" ); // we don't scan
directories, only files

KnowledgeAgent kagent = KnowledgeAgentFactory.newKnowledgeAgent( "test agent",
kaconf );
// resource to the change-set xml for the resources to add
kagent.applyChangeSet( ResourceFactory.newUrlResource( url ) );
```

Este es un ejemplo de un **`change-set.xml`**.

```
<change-set xmlns='http://drools.org/drools-5.0/change-set'";
    xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'
    xs:schemaLocation='http://drools.org/drools-5.0/change-set drools-change-set-
5.0.xsd' >
    <add>
        <resource source='http://localhost:9000/TEST.pkg' type='PKG' />
    </add>
</change-set>
```

El escaneo de recursos está habilitado por defecto. Es un servicio y se debe iniciar y los mismo aplica para la notificación. Esto se puede lograr por medio de ResourceFactory.

```
ResourceFactory.getResourceChangeNotifierService().start();
ResourceFactory.getResourceChangeScannerService().start();
```

Después se muestra la pantalla de implementación de la interfaz del usuario BRMS, la cual proporciona las URLs y las descargas de los paquetes.

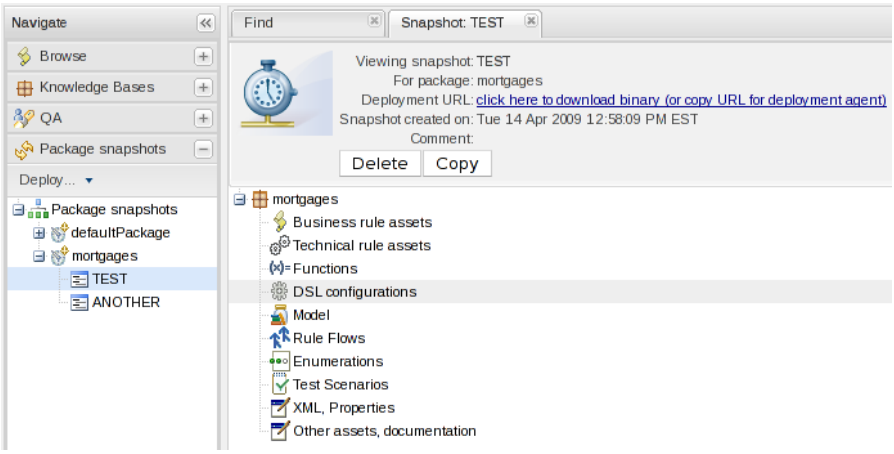


Figura 8.1. Implementación de tomas de pantalla

Puede ver la **`Package URI`**. Esta es la URL que necesita incluir en el archivo **`change-set.xml`** para especificar que quiere este paquete. Especifica una versión exacta, en este caso una toma de pantalla. Cada toma de pantalla tiene su propia URL. Si quiere la última versión entonces reemplace **`NewSnapshot`** con **`LATEST`**.

También puede descargar un archivo de paquete (**`PKG`**) de aquí. Ponga ese archivo en un directorio y use la funcionalidad **`file`** o **`dir`** del KnowledgeAgent. Esto contactará de manera automática el servidor JBoss Enterprise BRMS Platform para las actualizaciones que no sean deseadas en algunos escenarios.

8.2. Implementación manual

Esta sección es para usuarios avanzados que estén integrando la implementación en su propio mecanismo. Normalmente debe utilizar el agente de conocimiento.

Para aquellos quienes no desean usar la implementación automática del KnowledgeAgent, la implementación manual es bastante simple. Los paquetes binarios que JBoss Enterprise BRMS Platform produce son objetos de paquetes serializados. Puede des-serializarlos y agregarlos a cualquier KnowledgeBase.

Desde JBoss Enterprise BRMS Platform, los paquetes binarios se proporcionan desde la última versión de un paquete (una vez que el paquete se ha validado de manera exitosa y se ha construído) o desde las tomas de pantalla de la implementación. Las URLs que JBoss Enterprise BRMS Platform expone proporcionan el paquete binario usando HTTP. También puede emitir un comando "HEAD" para ver la última vez que se actualizó un paquete.

8.3. WebDAV

El repositorio de segundo palno también se puede acceder por medio de **`WebDav`**. WebDAV es una interfaz de programación de aplicaciones del sistema de archivos basado en HTTP. La mayoría de los sistemas operativos modernos incluyendo Microsoft Windows, Apple MacOS X y Linux proporcionan soporte integrado para acceder WebDav. Consulte la documentación de su sistema operativo para ver las instrucciones de configuración. También hay muchos clientes WebDav de terceros disponibles para la mayoría de las plataformas.

La URL para acceder su repositorio usando WebDav es casi la misma que la de la interfaz web. Simplemente reemplace **`Guvnor.html`** al final con **`/webdav/`** así como lo muestra este ejemplo:

```
http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav/
```

Se requiere autenticación como siempre. WebDav proporciona un directorio de paquetes y tomas de pantalla. El directorio **`Snapshots`** es de sólo lectura y es esencialmente una vista de las tomas de pantalla creadas de los paquetes de conocimiento. El directorio de paquetes contiene una lista de paquetes de conocimiento en el repositorio como directorios que a su vez contienen los activos individuales como archivos.

8.3.1. WebDav y caracteres especiales

BRMS soporta caracteres UTF-8 como parte de los nombres de las reglas, sin embargo, cuando las reglas se copian por medio de WebDav los caracteres multibyte se decodifican como ISO-8859-1.

Red Hat no recomienda el usar caracteres especiales en nombres de reglas; sin embargo, si se utilizan caracteres especiales entonces el Web Connector se debe cambiar para que soporte Unicode.

Para agregar soporte para Unicode complete los siguientes pasos:

Procedimiento 8.1. Agregue soporte Unicode

1. Detenga el servidor de aplicaciones.
2. Abra el archivo **`server.xml`**. Este archivo se encuentra en el directorio **`jbossweb.sar`**.
3. Agregue ***`URIEncoding="UTF-8"`*** al conector web. Por ejemplo, para el código HTTP el código debe ser el siguiente:

   ```
   <Connector protocol="HTTP/1.1" port="8080" address="${jboss.bind.address}"
   connectionTimeout="20000" redirectPort="8443" URIEncoding="UTF-8" />
   ```

4. Inicie el servidor de aplicaciones

8.4. URLs

La URL de implementación del paquete mencionada en la sección sobre el agente de conocimiento también tiene otras funcionalidades.

Para obtener el DRL generado para ese paquete en lugar del paquete binario agregue **`.drl`** al final de

una URL, por ejemplo, /package/testPDSGetPackage/LATEST.drl Al agregar **`/assetName.drl`** se podrá ver el DRL generado para ese objeto incluso si no es un archivo DRL tal como **`/package/testPDSGetPackage/LATEST/SomeFile.drl`**.

Capítulo 9. Buzón de entrada y comentarios

Importante

Disponible sólo en JBoss Enterprise BRMS Platform versión 5.1.0 y posteriores.

Las funcionalidades del buzón de entrada y los comentarios ayudan a los usuarios a administrar cambios en los objetos proporcionando una fuente adicional de documentación para los objetos y un sistema de notificación para los cambios en los objetos.

9.1. Comentarios

Cualquier usuario puede agregar un comentario a cualquier objeto en la sección de comentarios a continuación debajo de la casilla de documentación del objeto. Cada comentario se registra junto con la identidad del usuario que realiza el comentario y la fecha y hora en que se realizó el comentario. Los administradores pueden borrar todos los comentario en un objeto dado, pero los otros usuarios solo pueden agregar comentarios.

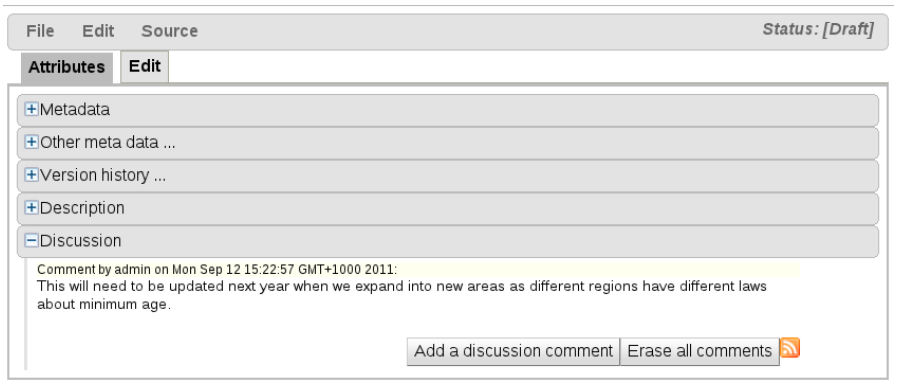


Figura 9.1. Comentarios

9.2. Buzón de entrada

El "buzón de entrada" se encuentra en el área de navegación en el panel de navegación. Brinda acceso rápido a los objetos en los que el usuario ha trabajado recientemente así como las notificaciones de los cambios a los objetos que el usuario modificó anteriormente.

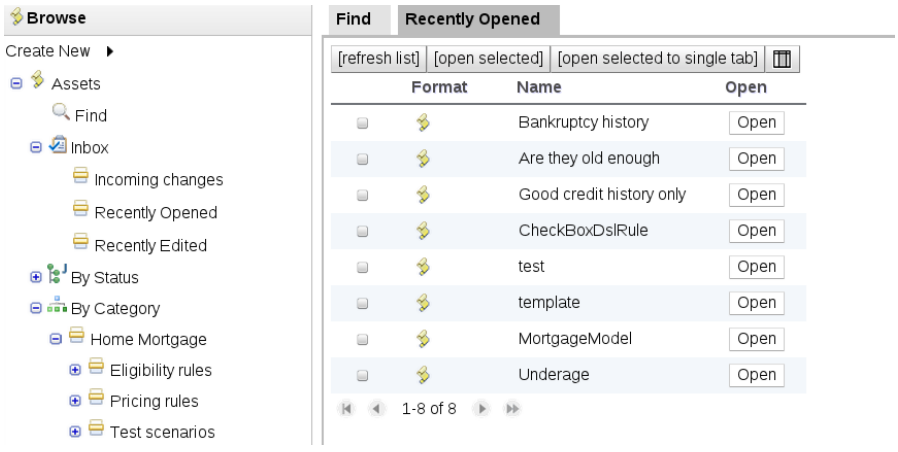


Figura 9.2. Buzón de entrada

El buzón de entrada tiene tres "sub-buzones":

Cambios entrantes

Aquí se listan los cambios que un usuario haya realizado en un objeto.

Abierto recientemente

Aquí se listan los últimos 100 objetos que el usuario ha abierto para tener acceso rápido.

Modificado recientemente

Aquí se listan los últimos 100 objetos que el usuario ha modificado para facilitar el acceso rápido.

Capítulo 10. Integración de JBoss Developer Studio

EGT (del inglés Eclipse Guvnor Tools) proporciona una interfaz para que los desarrolladores puedan leer, escribir, agregar y borrar activos de un servidor JBoss Enterprise BRMS Platform usando JBoss Developer Studio 4. EGT le brinda a los desarrolladores una interfaz similar a los sistemas de control tradicionales tal como Subversion.

Importante

JBoss solo proporciona soporte para esta funcionalidad en JBoss Developer Studio versiones 4.

El repositorio de BRMS Platform y EGT no tienen la intención de reemplazar un sistema de control fuente dedicado en su entorno de desarrollo sino que brinda una manera conveniente de acceso para los desarrolladores.

Importante

Guvnor es el nombre del proyecto de código abierto sobre el cual se ha construído JBoss Enterprise BRMS Platform. Eclipse Guvnor Tools se desarrollaron con esto en mente. En cualquier lugar en donde EGT se refiera a "Guvnor" se puede considerar como equivalente a "BRMS Platform". Para evitar confusión en lo que queda de este capítulo utilizaremos el término "Guvnor".

10.1. Sinopsis de las funcionalidades

EGT tiene dos vistas – Explorador del repositorio e historial de versiones – son el punto central de gran parte de la interacción con el repositorio Guvnor.

La perspectiva "Guvnor Repository Exploring" se brinda como la distribución sugerida. Se puede acceder desde la ventana **Open Perspective** (**Window** -> **Open Perspective** -> **Other...**).

En el lado izquierdo se encuentran las vistas **Guvnor Repository Explorer** y **Eclipse Properties**, la vista **Guvnor Resource History** se encuentra en la parte inferior y el **Eclipse Resource Navigator** está en el lado derecho. El **Guvnor Repository Explorer** propociona acceso a los recursos del repositorio Guvnor en un formato de árbol navegable. La vista **Guvnor Resource History** muestra revisiones de recursos especificos disponibles en el repositorio.

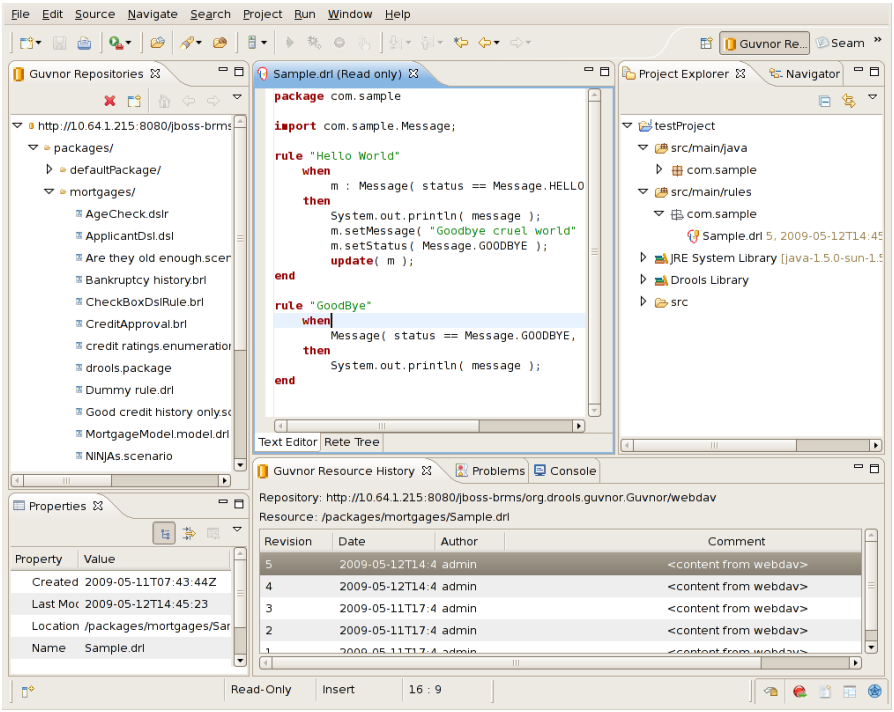


Figura 10.1. Perspectivas de "exploración del repositorio Guvnor"

10.2. Asistente de conexión Guvnor

Después de abrir la perspectiva Guvnor, la primera tarea es realizar una conexión a un repositorio Guvnor. Esto lo maneja el asistente de conexión Guvnor. Este asistente aparece en varios lugares dentro de EGT (como se detalla a continuación), pero en esta sección vamos a abordar solo los dos puntos de entrada más básicos. El asistente de conexión Guvnor se puede iniciar utilizando el menú de Eclipse: File , New , Other , Guvnor , Guvnor repository location o en el explorador Guvnor utilizando el menú desplegable:

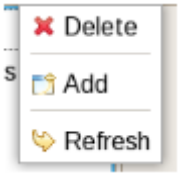


Figura 10.2. Asistente de conexión

o el botón en el menú:

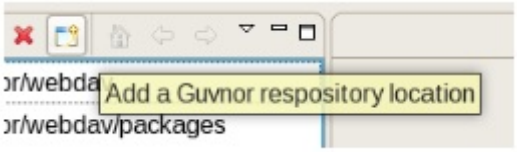


Figura 10.3. Asistente de conexión

El seleccionar cualquiera de estos iniciará el asistente de conexión Guvnor:

Figura 10.4. Asistente de conexión

Los valores predeterminados aparecen en los campos de ubicación, puerto y repositorio, (consulte la sección "Preferencias para Guvnor" a continuación para obtener mayores detalles sobre cómo cambiar estos valores predeterminados).

Por suspuesto, cualquiera de estos campos se puede modificar escribiendo en la casilla correspondiente. Arrastre y suelte o peque en el campo de la ubicación de un repositorio Guvnor típico URL tal como: **`http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav`**

Los resultados en la URL se analizan sintácticamente en los campos respectivos también. La información de autenticación (nombre del usuario y contraseña) se pueden almacenar opcionalmente en el archivo clave del banco de trabajo de Eclipse basado en la selección de "Guardar nombre de usuario y contraseña". Si la información no se almacena en el key-ring entonces el EGT usa la autenticación de sesión. Esto significa que las credenciales proporcionadas se utilizan solamente durante el tiempo de vida de la instancia del banco de trabajo de Eclipse.

Si la información de autenticación no se almacena en el key-ring o si no es válida entonces el EGT le preguntará cuando tenga que acceder al respositorio Guvnor:

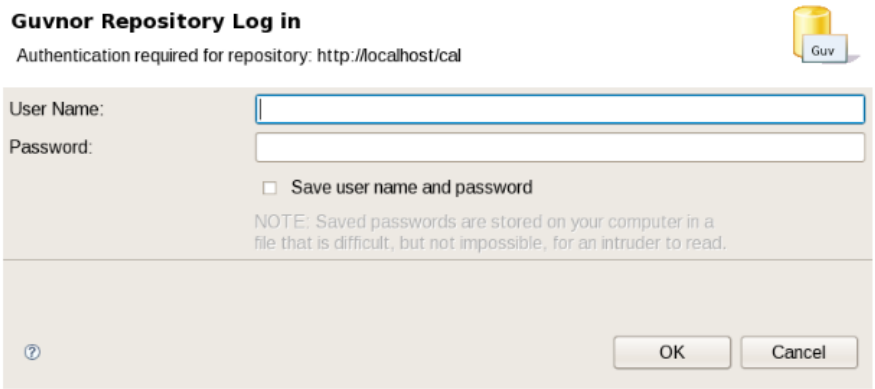

Guvnor Repository Log in
Authentication required for repository: http://localhost/cal
Guv
User Name:
Password:
Save user name and password
NOTE: Saved passwords are stored on your computer in a file that is difficult, but not impossible, for an intruder to read.
OK
Cancel

Figura 10.5. Inicio de sesión

Si la autenticación falla entonces el EGT volverá a intentar una vez más y luego emitirá un error de fallo de autenticación. Si reste error se presenta entonces puede volver a intentar la misma operación y brindar información de autenticación diferente. El EGT llama al repositorio Guvnor varias veces como cuando determina si hay actualizaciones de los recursos así que si utiliza autenticación de sesión entonces la ventana de autenticación aparecerá en momentos diferentes durante la sesión de Eclipse, dependiendo de las acciones que tome. Para facilitar el uso recomendamos el guardar la información de autenticación en la clave de Eclipse. El archivo de clave de Eclipse es diferente de los archivos de claves que se encuentran en algunas plataformas tal como Mac OS X y muchas formas de Linux. Por lo tanto algunas veces si accede un repositorio Guvnor por fuera del EGT, los archivos de claves pueden des-sincronizarse y se le pedirá que se autentique en Eclipse. Es una molesta pero sus credenciales usuales aplican aquí.

Una vez que se completa el asistente de conexión Guvnor aparecerá la nueva conexión del repositorio en **`Guvnor Repository Explorer`**. Luego puede expandir la vista de árbol para ver el contenido del repositorio.

10.3. Explorador del repositorio Guvnor

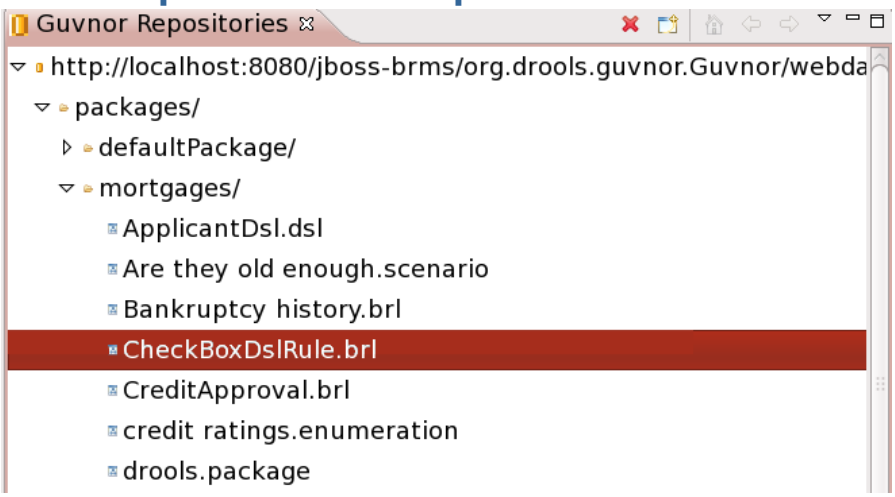


Figura 10.6. Explorador

La vista del explorador del repositorio Guvnor contiene estructuras de árbol para el contenido del repositorio Guvnor. Tal como se describió anteriormente hay acciones del menú y de la barra de herramientas para crear conexiones del repositorio Guvnor. La “X” roja en la barra de herramientas y “Delete” en el menú borra una conexión del repositorio Guvnor y la opción “Refresh” del menú vuelve a cargar el contenido del árbol para el nodo seleccionado. Finalmente hay un número de opciones en la barra de herramientas y en el menú para soportar la funcionalidad “drill-into”: en la barra de herramientas estas se representan por la casa (“return to top level/home”) y las flechas (go into/back). Drill-down es útil al trabajar con estructuras de árboles profundos y cuando desea concentrarse en una sola rama del árbol.

Hay un número de operaciones que se pueden realizar en los archivos del repositorio Guvnor. El seleccionar un archivo en el repositorio Guvnor hace que la vista de propiedades de Eclipse se actualice con los detalles de ese archivo:

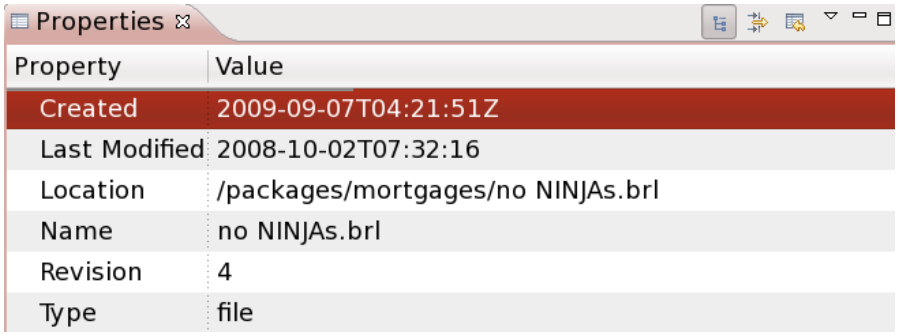


Figura 10.7. Propiedades

El hacer doble clic en una carpeta (directorio) en el árbol hará que ese directorio se expanda si está plegado o viceversa. El hacer doble clic en un archivo en el árbol hará que se abrá un editor de sólo lectura en Eclipse, mostrando el contenido de ese archivo:

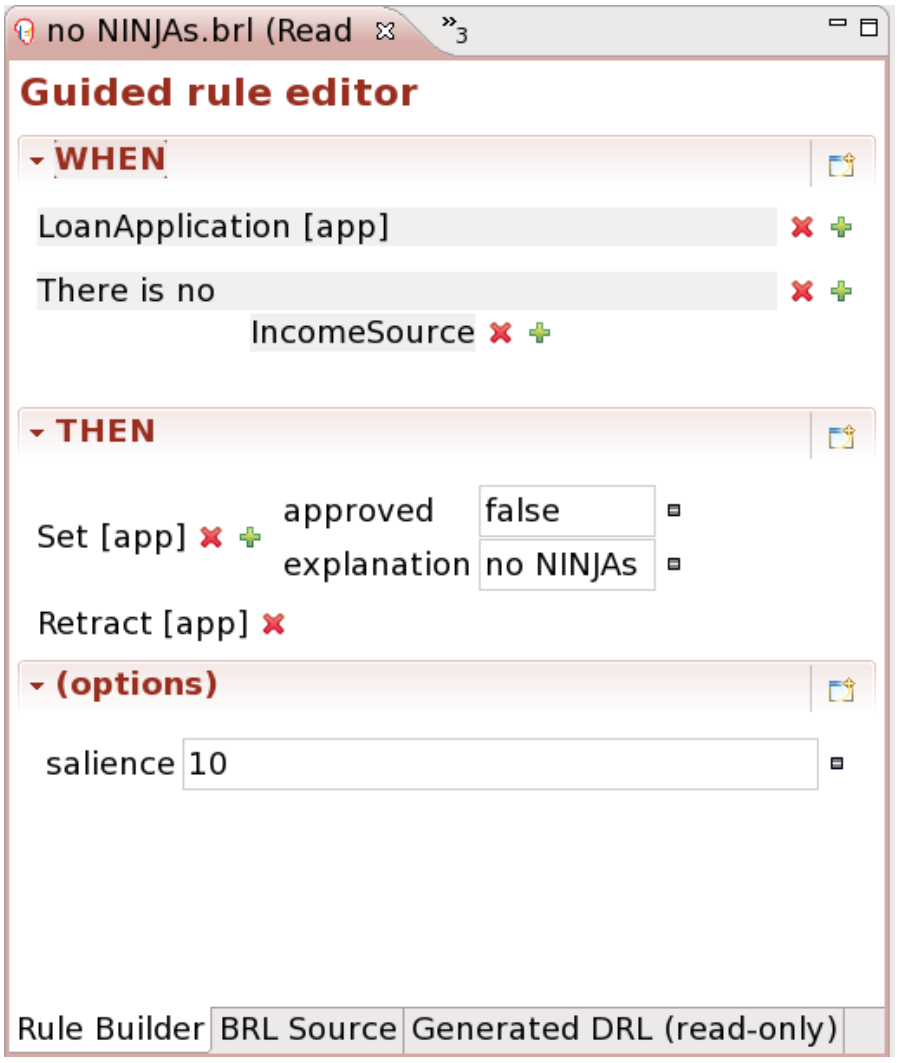


Figura 10.8. Contenido de archivos

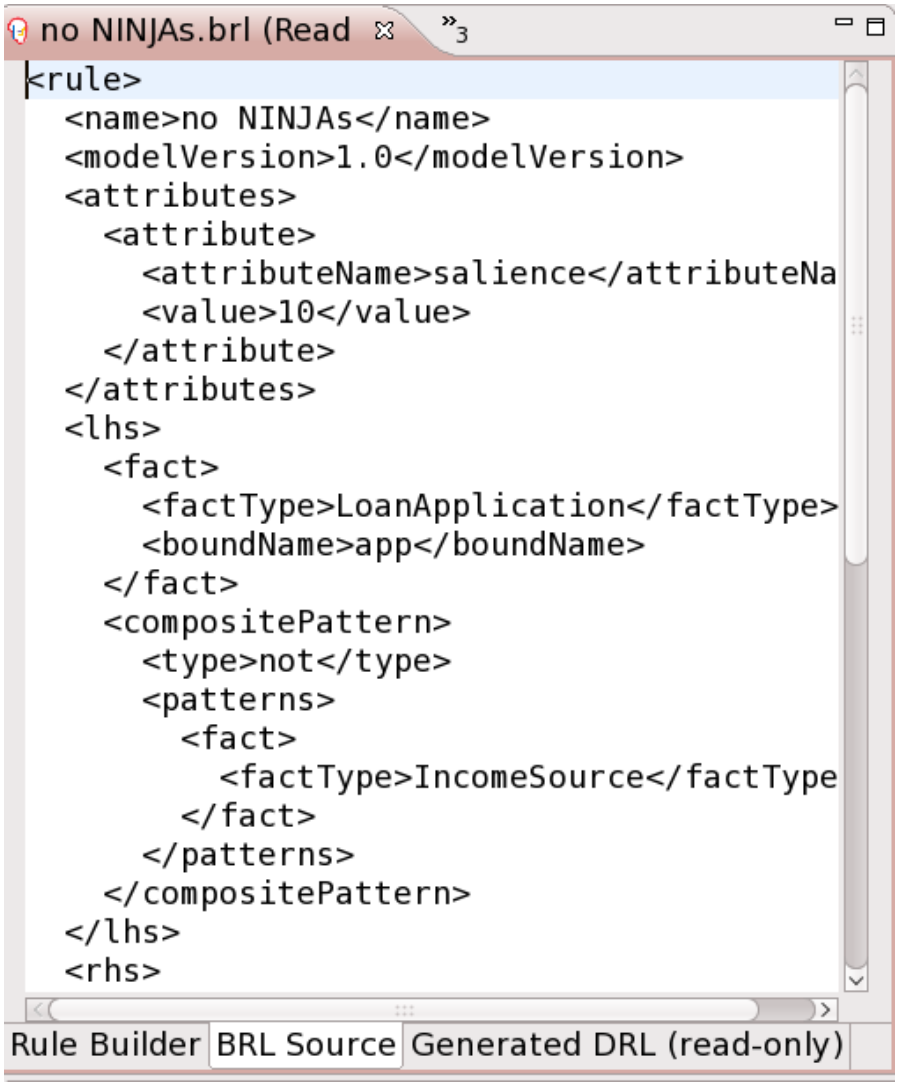


Figura 10.9. Contenido de archivos

El arrastrar un archivo del árbol del repositorio Guvnor en una carpeta en un proyecto local de Eclipse (por ejemplo en la vista del navegador de recursos de Eclipse) hará que se realice una copia de ese archivo en el espacio de trabajo local de Eclipse, (nota: también puede utilizar “Save As…” cuando un archivo está abierto en un editor de sólo lectura para guardar una copia local escribible del contenido. Sin embargo, al hacer esto no se asociará el archivo creado con su fuente Guvnor). Finalmente puede ver el historial de revisiones de un archivo seleccionado en el árbol usando la opción “Show History” del menú de contexto, (los detalles del historial del recursos se discute a continuación).

10.4. Copias locales de archivos Guvnor

Como lo mencionamos en la Introducción, el propósito principal del EGT es el permitir el desarrollo usando recursos que se mantienen en un repositorio Guvnor. Hay dos métodos de obtener copias locales de recursos del repositorio Guvnor:

1. Arrastre y suelte del explorador del repositorio Guvnor tal como se describió anteriormente.

2. Usando el asistente “import from Guvnor” tal como se explicó en Sección 10.6, “Importación de recursos del repositorio Guvnor”

Cuando se crean copias locales de los archivos del repositorio Guvnor, el EGT establece una

asociación entre la copia local y el archivo maestro en el repositorio, (esta información se mantiene en el directorio “.guvnorinfo” (normalmente) escondido en el proyecto local y como todos los metadatos los usuarios no lo deben cambiar). Esta asociación permite operaciones tal como la actualización y el guardar cambios en sincronización con la copia maestra que se mantiene en el repositorio Guvnor. El EGT decora los recursos locales asociados con copias maestras del repositorio Guvnor. Esta decoración aparece en las vistas de Eclipse que cumplen con los requerimientos del navegador común de Eclipse tal como el navegador de recursos de Eclipse y el explorador de paquetes Java. La imagen a continuación muestra la decoración en el navegador de recursos de Eclipse:

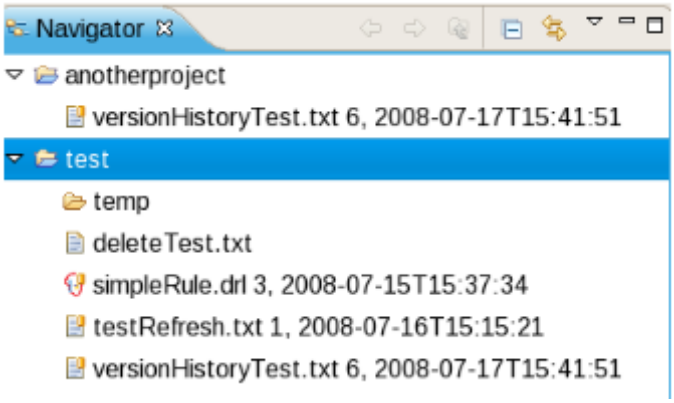


Figura 10.10. Navegador

Note el ícono Guvnor en la parte superior derecha de las imágenes de archivo y los detalles de revisión de Guvnor que se agregan a los nombres de los archivos, (se puede cambiar la presencia/ubicación de estos, consulte “Guvnor Preferences” a continuación para obtener mayores detalles). Por ejemplo, aquí vemos que “simpleRule.drl” está asociado con un recurso del repositorio Guvnor y la copia local se basa en la revisión 3 con un sello de fecha 7-15-2008, 15:37:34. Sin embargo, el archivo “deleteTest.txt,” no está asociado con un archivo del repositorio Guvnor. Puede encontrar mayores detalles sobre la asociación en la pagina de propiedades estándar de Eclipse por medio de la opción "Properties" del menú de contexto:

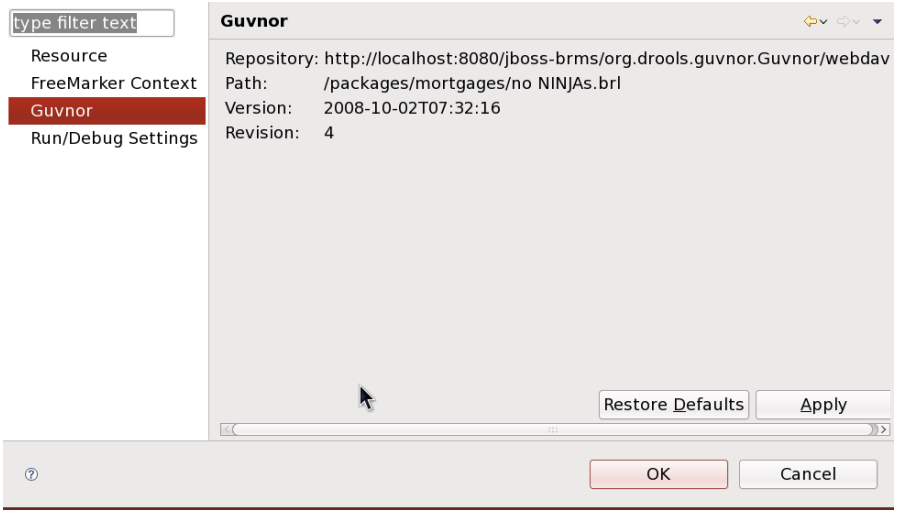


Figura 10.11. Propiedades

El EGT contribuye una página de propiedades a la ventana de propiedades de Eclipse, arriba puede ver su contenido. Se puede ver el repositorio especifico Guvnor, la ubicación dentro del repositorio, la versión (fecha/sello de fecha) y el número de la revisión.

10.5. Acciones para recursos locales Guvnor

El EGT proporciona un número de acciones (disponibles por medio del menú de contexto de “Guvnor” en los archivos) para trabajar con archivos, ambos asociados con las copias maestras del repositorio Guvnor y las que no están asociadas. Las acciones son:

- Actualización
- Agregar
- Guardar cambios
- Mostrar historial
- Comparar con la versión
- Cambiar a la versión
- Borrar
- Desconectar

Cada una de estas acciones se describirán a continuación.

Actualizar acción:

La acción de actualización está disponible para uno o más recursos Guvnor que no están en sincronización con las copias maestras del repositorio Guvnor. Estos recursos no estarían en sincronización ya que (una de las siguientes o ambas): (1) hay cambios locales en estos recursos o (2) las copias maestras han cambiado en el repositorio Guvnor. Al realizar la acción de actualización se reemplaza el contenido del archivo local con el contenido actual de las copias maestras del repositorio Guvnor (equivalente a “cambio de versión” para la última versión).

Agregar acción:

La acción de agregar está disponible para uno o más archivos locales que no estén asociados con una copia maestra del repositorio Guvnor. El seleccionar la acción de agregar inicia el asistente “Add to Guvnor”:

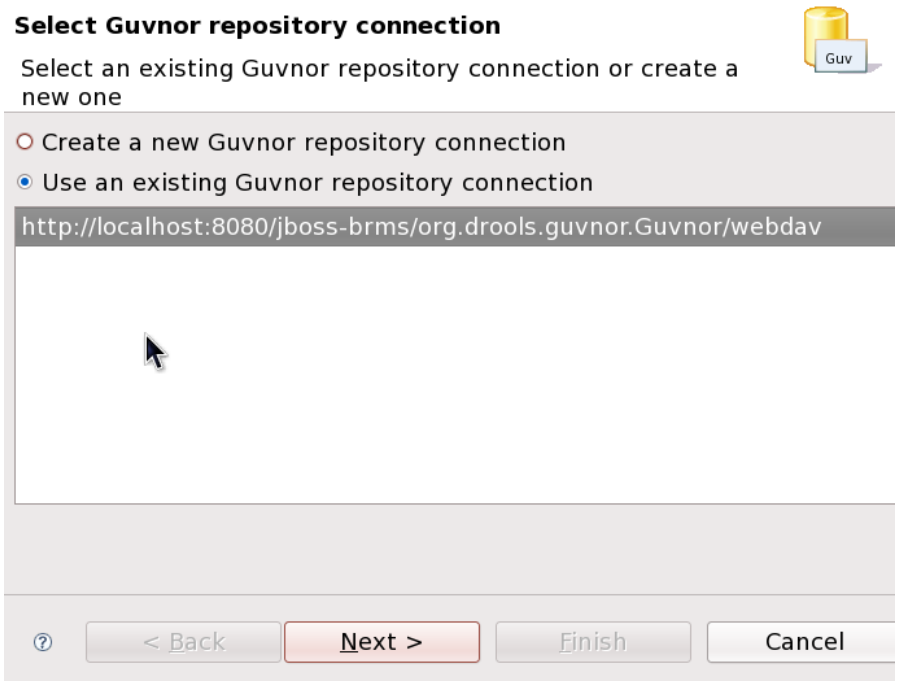


Figura 10.12. Agregar acción

La primera página del asistente le pide que seleccione el repositorio Guvnor de destino y le da la opción de crear una nueva conexión del repositorio Guvnor (en cuyo caso la segunda página es la misma del asistente de conexiones de Guvnor que describimos anteriormente). Una vez seleccione el repositorio Guvnor de destino, el asistente le pide la ubicación de la carpeta para agregar los archivos seleccionados:

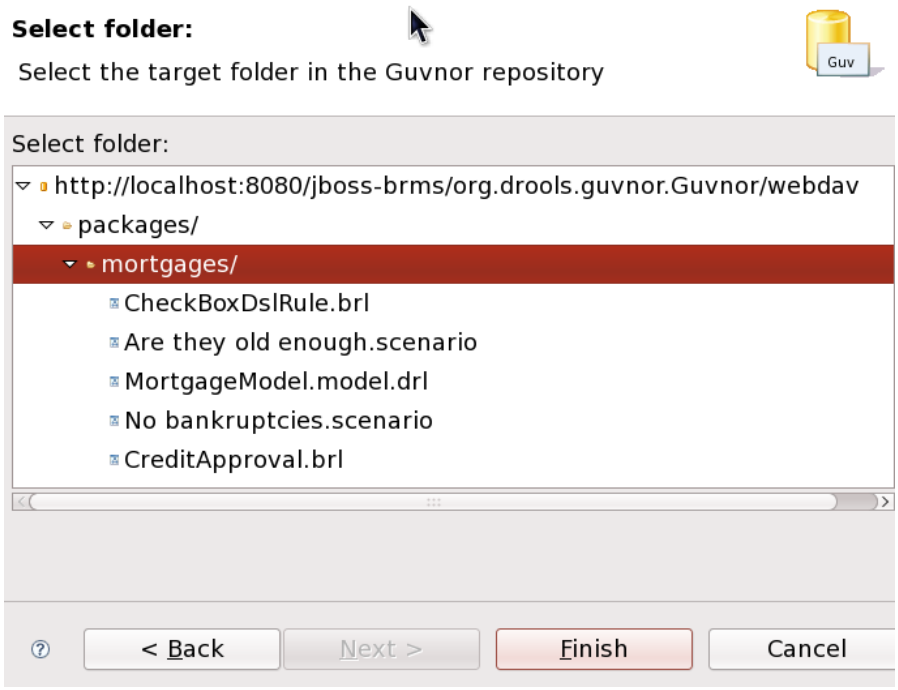


Figura 10.13. Agregar acción

Aquí seleccioné la carpeta **`mortages`** como el destino. El hacer clic en “Finish” agrega los archivos seleccionados al repositorio Guvnor y crea una asociación entre los archivos locales y los del repositorio Guvnor. El asistente no le permitirá sobreescribir archivos existentes del repositorio Guvnor al agregar nuevos archivos.

Compare con la acción de versión:

La acción de comparación con versión está habilitada para un archivo asociado del repositorio Guvnor. Esta acción primero abre un asistente que le pide la versión para realizar la comparación (con el contenido del archivo local):

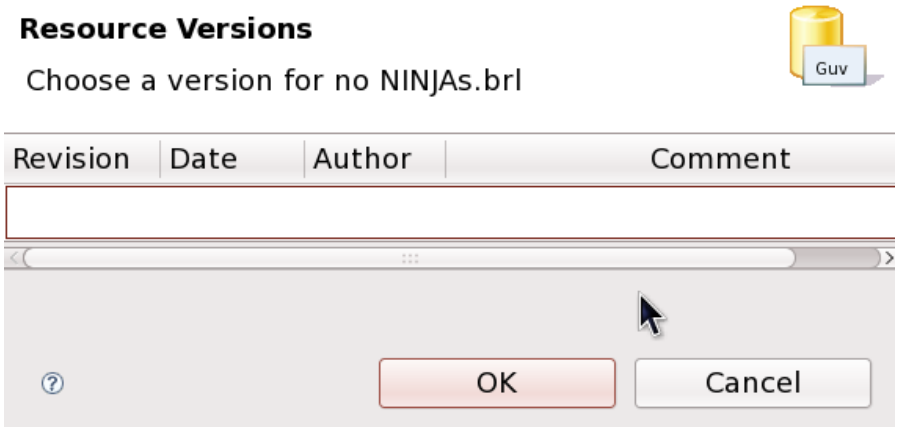


Figura 10.14. Comparación

Una vez que se escoge la revisión entonces la acción abre el editor de comparación de Eclipse (sólo lectura):

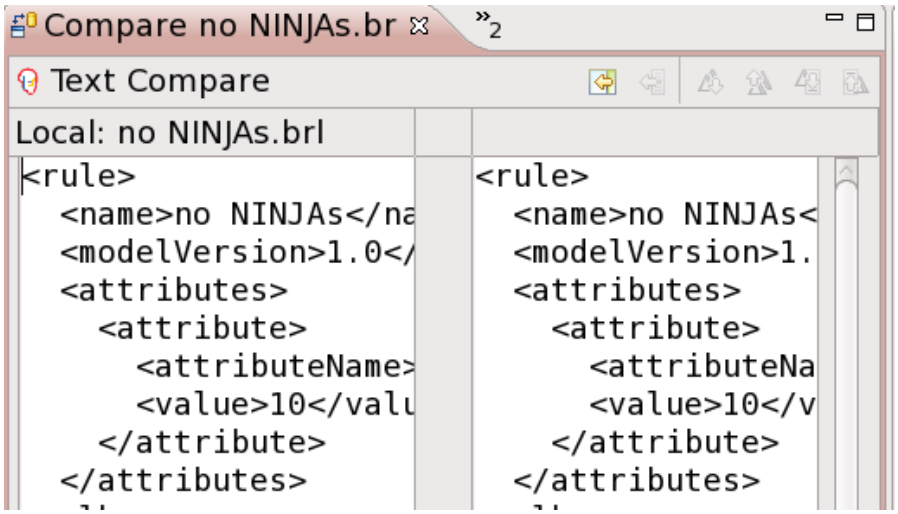


Figura 10.15. Comparación

Este editor usa las técnicas de comparación estándares de Eclipse para mostrar las diferencias en las dos versiones. En los casos en donde no hay diferencias, el editor no abrirá: en lugar, aparecerá una ventana diciendo que no hay diferencias.

Cambio a la acción de versión:

La acción de cambio de versión está habilitada para un archivo asociado del repositorio Guvnor. Primero la acción de cambio de versión le pide que seleccione una versión:

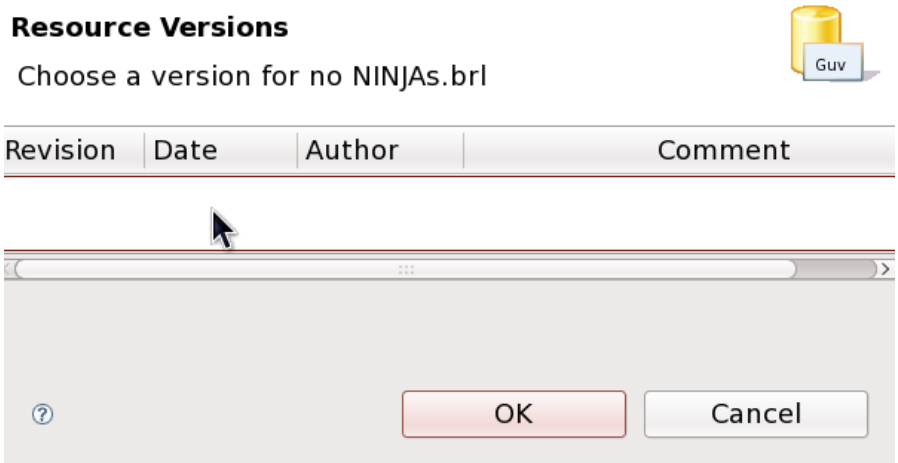


Figura 10.16. Versiones

Una vez que se escoja la versión, la acción de cambio de versión reemplaza el contenido del archivo local con el de la revisión seleccionada.

Acción de borrar:

La acción de borrar está habilitada para uno o más archivos asociados del repositorio Guvnor. Después

de confirmar por medio de la ventana, la acción de borrar elimina los archivos en el repositorio Guvnor y borra los metadatos locales para la asociación del repositorio Guvnor.

Acción de desconexión:

La acción de desconexión está habilitada para uno o más archivos asociados del repositorio Guvnor y borra los metadatos locales para la asociación del repositorio Guvnor.

Vista del historial de recursos Guvnor:

La vista del historial de recursos Guvnor muestra los detalles del historial de revisión para los archivos seleccionados tanto locales como los que se encuentran en los repositorios Guvnor. El estado inicial de esta vista es:

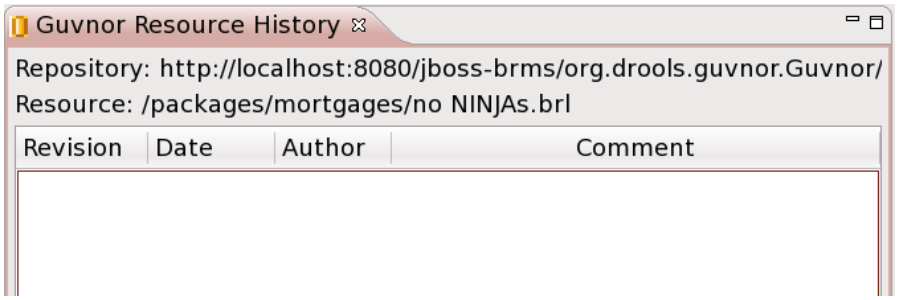


Figura 10.17. Historial

La vista del historial de recursos Guvnor se llena por medio de las acciones "Show History" en el menú de contexto "Guvnor" local o en el menú de contexto para un archivo del repositorio Guvnor en el explorador del repositorio Guvnor. Una vez que se realiza esta acción, la vista del historial de recursos Guvnor se actualiza para mostrar el historial de revisiones:

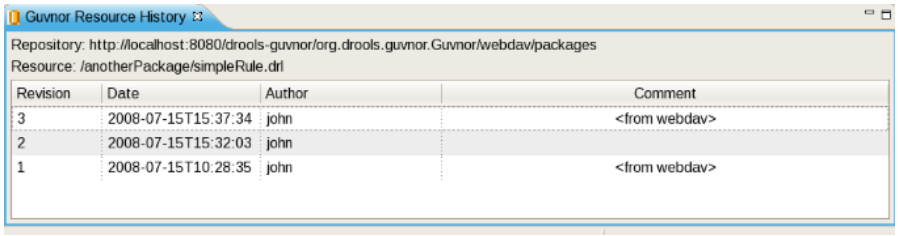


Figura 10.18. Historial

Aquí vemos que el archivo “simpleRule.drl” tiene tres revisiones. Al hacer doble clic en una fila de revisión (o menú de contexto “Open (Read only)”) abre un editor de solo lectura en Eclipse con el contenido de la revisión, (nota: también puede guardar con “Save As...” cuando un archivo está abierto en un editor de sólo lectura para guardar una copia escribible local del contenido. Sin embargo, al hacer esto, no se asociará el archivo creado con su fuente Guvnor).

10.6. Importación de recursos del repositorio Guvnor

Además del archivo de arrastre y suelte de la vista del explorador del repositorio Guvnor, EGT también incluye un asistente para copiar uno o más archivos desde un repositorio Guvnor al espacio de trabajo local (y establecer la asociación con el repositorio Guvnor). Este asistente está disponible desde las opciones del menú de Eclipse Import , Guvnor, Resource from Guvnor y Eclipse File, New, Other, Guvnor, Resource from Guvnor, (nota: el asistentes es idéntico pero aparece en ambos lugares para

acomodar aquellos usuarios que tienden a ver esta funcionalidad como si estuviera en una sola categoría). La primera página del asistente le pide que escoja el repositorio Guvnor fuente y le da la opción de crear una nueva conexión del repositorio Guvnor (en curo caso la segunda página es la misma que el asistente de conexión de Guvnor descrito anteriormente).

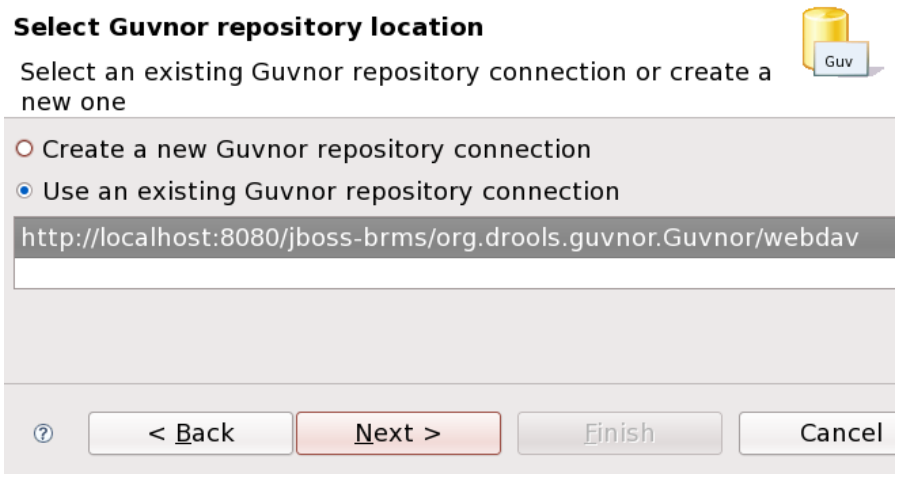


Figura 10.19. Importación

Una vez que se selecciona el repositorio Guvnor fuente, el asistente le pide que escoja el recurso:

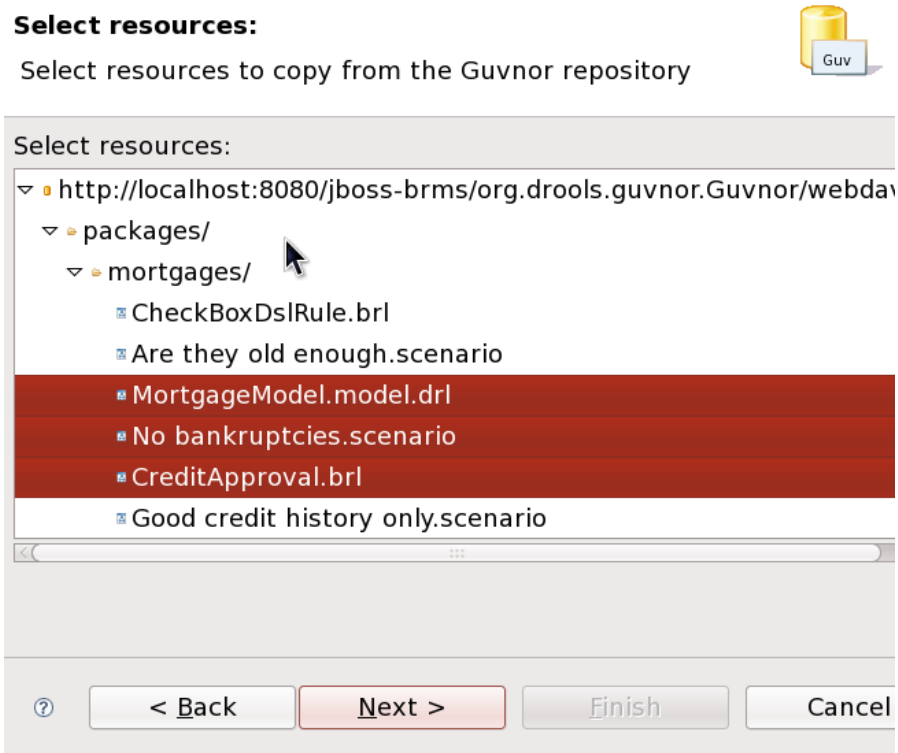


Figura 10.20. Importación

Finalmente se escoge la ubicación destino en el espacio de trabajo local:

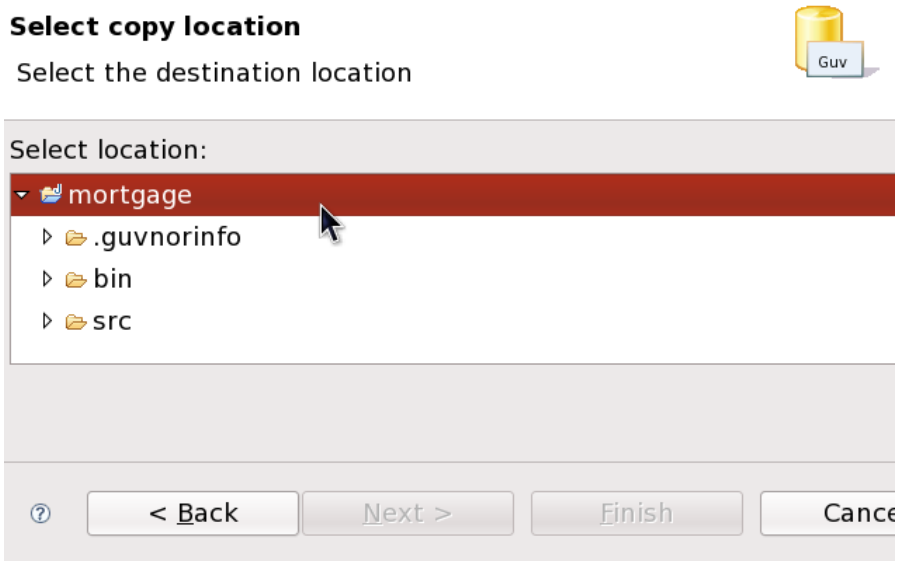


Figura 10.21. Importación

Una vez termine, el asistente copia los archivos seleccionados del repositorio Guvnor al espacio de trabajo local. Si ya existe un archivo con el mismo nombre en el destino entonces el asistente usa la ventana estándar de Eclipse “prompt for rename”:

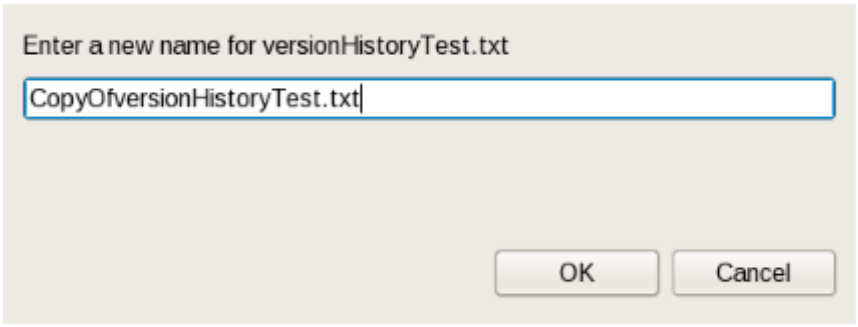


Figura 10.22. Copiar

10.7. Preferencias plug-in de Guvnor

EGT brinda una página de preferencias en la categoría “Guvnor”:

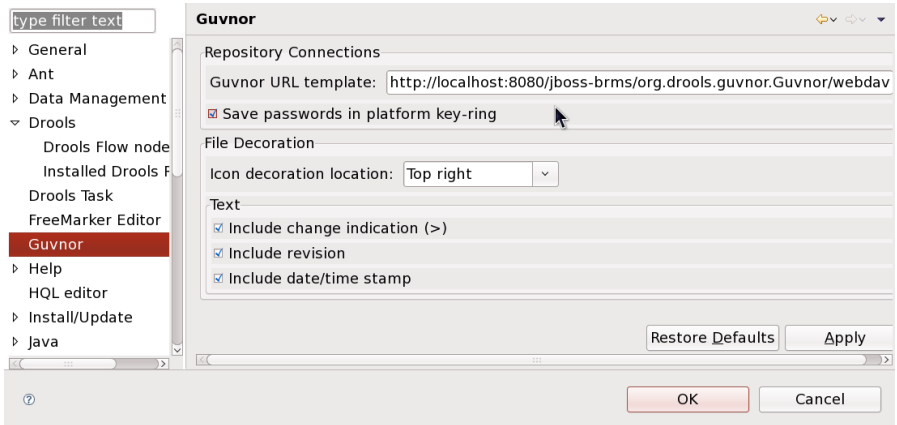


Figura 10.23. Preferencias

Las preferencias cubren dos categorías: las conexiones del repositorio Guvnor y las decoraciones de recursos del repositorio local Guvnor.

Preferencias de conexión del repositorio Guvnor

Hay dos preferencias que se pueden establecer para las conexiones del repositorio Guvnor y estas se utilizan al crear nuevas conexiones. La primera es una plantilla URL del repositorio Guvnor predeterminado, la cual facilita crear múltiples conexiones similares simplemente cambiando parte del campo tal como el nombre del host. La segunda que por defecto se debe habilitar el guardar la información de autenticación en la clave de la plataforma de Eclipse. Así como con la plantilla URL del repositorio Guvnor se puede determinar si el guardar una instancia especifica de la información de autenticación en la clave de la plataforma Eclipse al crear la conexión. Es decir ambas preferencias son simplemente valores convenientes establecidos con valores predeterminados razonables.

Preferencias de decoración de recursos del repositorio Guvnor local

La segunda categoría de preferencias que EGT proporciona se encarga de la manera en que se presenta la decoración de recursos locales asociados con recursos del repositorio Guvnor. Ya que el repositorio Guvnor no es un substituto para SCM y ya que las herramientas SCM en Eclipse tienden a decorar recursos locales, es útil el poder controlar la manera en que EGT decora sus recursos locales para evitar conflictos con paquetes SCM. En la sección “File Decoration” de la página de preferencias puede seleccionar la ubicación (superior izquierda o derecha o inferior izquierda o derecha) del ícono de decoración o puede escoger el no presentarla. En la sección “Text” puede formatear los metadatos Guvnor que se agregan a los nombres de archivos: si mostrar un indicador (>) cuando el archivo local tiene cambios que no se han guardado en el repositorio Guvnor. Si debe mostrar el número de la revisión, si debe mostrar el sello de fecha. Cualquier cambio a estas preferencias tienen efecto de manera inmediata al hacer clic en los botones “Apply” u “Ok”.

Capítulo 11. Configuración de clientes para la firma de paquetes de reglas

Lea esta sección para aprender sobre la firma de paquetes de reglas y la configuración del almacenamiento de llaves.

La *firma de paquetes de reglas* asegura que los paquetes de reglas no se puedan dañar o alterar durante la descarga del servidor de JBoss Enterprise BRMS Platform para las aplicaciones clientes. La firma de paquetes de reglas no está habilitada por defecto.

Importante

Red Hat recomienda que se habilite la firma de paquetes de reglas en entornos de producción.

La firma de paquetes se implementa usando la *criptografía de llaves públicas*. Se utiliza el comando JDK **`keytool`** para crear una llave privada y un certificado digital público correspondiente. Los paquetes firmados con una llave privada sólo se pueden verificar con el certificado que coincida. La llave privada se almacena en un archivo llamado *keystore* y el servidor lo utiliza para firmar automáticamente cada paquete. El certificado público se hace disponible para cada aplicación cliente en un almacén de llaves conocido como *almacén de confianza*. El certificado en el almacén de confianza se utiliza para verificar la autenticidad de los paquetes firmados. Los paquetes de reglas que están corruptos o que se han modificado durante la descarga serán rechazados por el cliente ya que la firma ya no coincidirá con el certificado.

El procedimiento a continuación describe el proceso de configuración de aplicaciones cliente para la firma de paquetes de reglas. Esto implica el configurar varias propiedades en el cliente. Las propiedades se puede establecer programáticamente usando el método **`System.setProperty`**. La clase **`org.drools.core.util.KeyStoreHelper`** contiene varias constantes que representan estas propiedades.

Antes de realizar esta tarea necesita:

- Un servidor JBoss Enterprise BRMS Platform ya instalado y configurado correctamente para la firma de paquetes de reglas.
- La URL para el almacén de confianza que contiene el certificado digital que el servidor JBoss Enterprise BRMS Platform utiliza.
- La contraseña para el almacén de confianza si hay una establecida.

Procedimiento 11.1. Configuración de clientes para la firma de paquetes de reglas

1. **Habilitar la firma**

   Configure la propiedad **`drools.serialization.sign`** como **`true`**.

   ```
   System.setProperty( KeyStoreHelper.PROP_SIGN, "true" );
   ```

2. **Configure la URL TrustStore**

   Configure la propiedad **`drools.serialization.public.keyStoreURL`** con la URL en donde se encuentra el TrustStore. Si el TrustStore se encuentra en la ruta de clase del cliente entonces esto se puede lograr usando el método **`getClass().getResource()`**.

Ejemplo 11.1. Cuando el TrustStore se encuentra en la ruta de clase del cliente

```
URL trustStoreURL = getClass().getResource( "BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

Ejemplo 11.2. Cuando el TrustStore se encuentra en un servidor web

```
URL trustStoreURL = new
URL("http://brms.intranet/resources/BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

Ejemplo 11.3. Cuando el TrustStore se encuentra en el sistema local de archivos

```
URL trustStoreURL = new URL("file:///mnt/fileserve/rules-
server/BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

3. **OPCIONAL: Establezca la contraseña del almacén de claves**

   Configure la propiedad **`drools.serialization.public.keyStorePwd`** con la contraseña para el almacén de confianza. Esto sólo se necesita si se requiere una contraseña para acceder el almacén de confianza.

   ```
   System.setProperty( KeyStoreHelper.PROP_PUB_KS_PWD, "sekretPasswordHere" );
   ```

- Consulte el manual del administrador BRMS para ver las instrucciones sobre cómo configurar el servidor para la firma de paquetes de reglas, http://docs.redhat.com/docs/en-US/JBoss_Enterprise_BRMS_Platform/5/html/BRMS_Administrator_Guide.
- Consulte http://en.wikipedia.org/wiki/Public-key_cryptography para obtener mayor información sobre la criptografía de claves públicas.

Historial de revisiones

Revisión 5.2.0-2.400	**2013-10-31**	**Rüdiger Landmann**

Rebuild with publican 4.0.0

Revisión 5.2.0-2	**2012-07-18**	**Anthony Towns**

Rebuild for Publican 3.0

Revisión 5.2.0-0	**August 4 2011**	**L Carlon**

Actualizado para 5.2.0
Se dividió el capítulo 3 (manual del usuario) en los capítulos 3, 4 y 5.
Se agregó información adicional sobre las tablas de decisión (capítulo 4.4.4).
Se agregó información adicional sobre el modelo de hechos (capítulo 5).
Se agregó un capítulo sobre grupos de trabajo (capítulo 6).

Revisión 5.1.0-0	**Mon December 13 2010**	**David Le Sage, Darrin Mison**

Actualizado para 5.1.0
El contenido sobre instalación y administración se movió al manual correspondiente.
Se agregó contenido para las nuevas funcionalidades del buzón de entrada y los comentarios (sección 3.9).
Se agregó contenido para la configuración de clientes para la funcionalidad de la firma de paquetes de reglas (capítulo 5).

Revisión 5.0.2-0	**Wed May 5 2010**	**Darrin Mison**

Actualizado para 5.0.2
BRMS-262 Se agregó contenido para la configuración de registros (sección 2.7)
BRMS-233 Se actualizaron varias tomas de pantalla (capítulo 4)

Revisión 5.0.1-0	**Fri Oct 3 2009**	**David Le Sage, Darrin Mison**

Actualizado para 5.0.1
BRMS-227 - Se agregó la sección 2.3 sobre localización

Revisión 5.0.0-1	**Thu 16 Jul 2009**	**Darrin Mison**

BRMS-203 - Se actulizaron los detalles para las bases de datos soportadas

Revisión 5.0.0-0	**Mon 18 May 2009**	**Darrin Mison**

Creado para 5.0.0